

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
10 May 2001 (10.05.2001)

PCT

(10) International Publication Number
WO 01/33381 A1

(51) International Patent Classification⁷: G06F 15/00,
15/16, 17/30, B41B 15/00

(21) International Application Number: PCT/US00/30536

(22) International Filing Date:
3 November 2000 (03.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/163,626 4 November 1999 (04.11.1999) US
09/570,583 12 May 2000 (12.05.2000) US

(71) Applicant (for all designated States except US): XDRIVE,
INC [US/US]; Attn: Legal Dept., 3002 Pennsylvania Av-
enue, Santa Monica, CA 90404 (US).

(72) Inventors; and

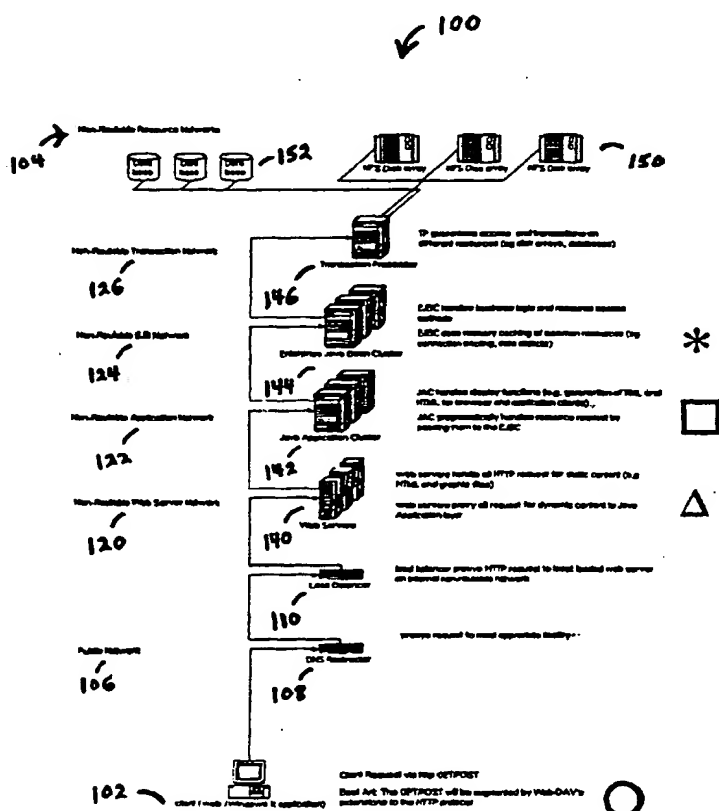
(75) Inventors/Applicants (for US only): O'BRIEN, Brett
[US/US]; XDrive, Inc., 3002 Pennsylvania Avenue, Santa
Monica, CA 90404 (US). WHITELEY, Sean [US/US];
XDrive, Inc., 3002 Pennsylvania Avenue, Santa Monica,
CA 90404 (US). MCGREGOR, Lucas [US/US]; XDrive,
Inc., 3002 Pennsylvania Avenue, Santa Monica, CA 90404
(US). HALD, Martin [DK/US]; XDrive, Inc., 3002
Pennsylvania Avenue, Santa Monica, CA 90404 (US).

(74) Agents: JORDAN, Andrew et al.; Cislo & Thomas LLP,
Suite 900, 233 Wilshire Boulevard, Santa Monica, CA
90401-1211 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU,
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ,
DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR,
HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR,
LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ,

[Continued on next page]

(54) Title: SHARED INTERNET STORAGE RESOURCE, USER INTERFACE SYSTEM, AND METHOD



(57) Abstract: The shared internet storage resources provides Internet-based file storage, retrieval, access, control, and manipulation for user. Additionally, an easy-to-use user interface is provided both for a browser or stand-alone application. The entire method provides means by which users can establish, use, and maintain files on the internet in a manner remote from their local computers yet in a manner that is similar to the file manipulation used on their local computers. A high capacity or other storage system is attached to the internet via an optional internet network that also serves to generate and direct metadata regarding the stored files. A web server (140) using a CGI, Java-based, or other interface transmits and retrieves TCP/IP packets or other internet information through a load balancer/firewall (110) by using XML to wrap the data packets. File instructions may be transmitted over the Internet to the Shared Resource System. The user's account may be password protected so that only the user may access his or her files. On the user's side, a stand-alone client application (142) or JavaScript object interpreted through a browser provide two means by which the XML or other markup language data stream may be received and put to use by the user. Internet-to-internet file transfer may be effected by directly downloading to the user's account space.

WO 01/33381 A1



NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) **Designated States (regional):** ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

- With international search report.
- Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SHARED INTERNET STORAGE RESOURCE, USER INTERFACE SYSTEM, AND METHOD

TECHNICAL FIELD

This invention relates to resources on computer networks, particularly the Internet, and more particularly to a file storage and retrieval system that is available worldwide via the Internet which additionally allows a direct transfer of Internet files to an Internet storage, retrieval, and sharing resource. The present invention acts in the manner of a "Internet hard disk" or "Internet hard drive" to provide online storage and retrieval resources for users.

BACKGROUND ART

The Internet is the worldwide computer network making available a vast number of computer and information resources to institutions and individuals. A significant part of the Internet is the worldwide web that allows for web pages to be written in HTML and transmitted upon demand throughout the Internet. Recent developments have better established the use of XML (Extensible Markup Language) as a subset of SGML (Standard Generalized Markup Language, ISO standard 8879:1986). FTP (File Transfer Protocol) provides means by which files may be transferred over the Internet. All of these protocols are generally well known in the art, and collateral resources can easily be obtained to describe these further.

Patents relevant to the present invention include U.S. Patent No. 5,771,354 issued to Christopher Crawford on June 23, 1998 for an Internet Online Backup System, and U.S. Patent No. 5,901,228 issued to Christopher Crawford on May 4, 1999 for a Commercial Online Backup Service.

Additionally, portable programming systems such as Java®, JavaBeans, and JavaScript have been extensively developed with an anticipation of future portability across the vast network that is the Internet. Java®-related systems allow for object-oriented programming whereby objects or "beans" allow the passing of self-contained modules with associated processing methods that are used to act upon the accompanying data. Consequently, the "bean" can travel through a network and, under appropriate circumstances, have certain processes activated allowing manipulation of the information contained in the bean.

Advancements in Java®-related systems have given rise to the Enterprise JavaBean™ (EJB). The Enterprise JavaBean™ allows for clustering of servers such that the bean is given independence from specific servers on the system, yet can be activated or "instantiated" such that error recovery is easier, the system as a whole is more robust, and processing of the bean can be performed asynchronously so that all events do not have to happen at a pre-set time or serially/one after the other.

Enterprise JavaBeans™/EJBs allow serialization of beans. Such serialization allows the bean to be represented as a data stream of determined length. In essence, this is just a data file that is interpreted in the proper context, much the same as any electronic information file. Such serialization of the EJB allows it to be replicated and stored in case of catastrophic failure of a preferred server or the like.

If the server upon which the instantiated EJB dies, goes down, or fails, a previously replicated twin can be used to continue the process and allow for error recovery. More information about Enterprise JavaBeans™ technology can be found in the white paper, "Enterprise JavaBeans™ Technology: Server Component Model for the Java™ Platform" by Anne Thomas, revised December 1998, prepared for Sun Microsystems, Inc. and published/made available by the Patricia Seybold Group of Boston, Massachusetts.

Due to the nature of new technologies, terms such as "bean" or "instantiated" may seem unfamiliar to those new

to the pertinent art. Reasons for this include the difficulty of communicating quickly new and complex subjects as well as the good-humored nature of those who intensely pursue the establishment of new technology, particularly software systems. Consequently, for Java®-related systems, a coffee theme is often present that indicates to those knowledgeable in the art the general subject matter of interest. While distinctions may be subtle in the art, they can be very important and serve the ends of those attempting to establish, share, and forward the technology.

Generally, home pages or other web pages are requested by the user through designation of the URL (Uniform Resource Locator). With the transmission to the user via TCP/IP protocol, the information present at the URL (and generally a file located somewhere on a computer) is transmitted to the user. The file may have links, or pointers, to other resources including images, graphics, audio or video streams, or other resources. Mark-up language is used on the Internet in an attempt to provide an open-ended structure by which information of any sort that can be stored electronically (or perhaps even otherwise) can be made available to an end user on demand. As such, the Internet is seen as a powerful tool making almost any information resource available to any computer or to any person using a computer.

Over the past several years, the personal computer has increased in power and capacity as commercial demand has driven the research and development of producers and vendors. It is now not uncommon to be able to easily find an Intel-manufactured 500 megahertz Pentium®-based system having well over 10 gigabytes of hard disk space, as well as 32 - 256 megabytes of RAM. As such, the power by which files may be received and acted upon by the local user through his or her PC has kept pace with the advances in technology.

However, there currently remain obstacles to universal access to an individual's own information stored on his or her computer. First of all, computers are very heavy. They are bulky. They generally weigh several kilograms and are not easily transportable. Lightweight laptop computers or the like generally do not have the same resources available to the user as a regular PC. Additionally, access to local area networks (LANs) is generally not available once the computer leaves the premises occupied by the LAN. Additionally, Internet access is often restricted by the use of a modem. Modems generally provide data transmission speeds on the order of 56 kilobits per second. This is approximately the same as 7 kilobytes per second. However, headers and other information are required to properly transmit information over the Internet and increase the effective size of files.

Even with the increased availability of broad band access to the Internet, it becomes an important feature of electronic information processing and the like in order to provide resident resources on the Internet. Such resources could include the sharing of files and the like in a manner that are easy to use and understand.

Due to these and other restrictions regarding data transport, transmission, and reception, a need has arisen for means by which files and other data may be available worldwide through the Internet and not tied to a local computer. The present invention addresses this demand by providing means by which files and other data may be stored on the Internet and made available worldwide through the Internet.

DISCLOSURE OF INVENTION

The present invention provides an "Internet hard drive" or "Internet hard disk" to and from which files may be stored and retrieved. Denominated commercially as "X:Drive," the present invention allows users to store files of foreseeably any type on a resource available throughout the Internet. Once available to the Internet, the files stored on the user's X:Drive are available to the same extent as the Internet, namely worldwide.

Note should be made that the term "X:Drive" refers both to the system as a whole and to the individual space allocated to an individual user. Consequently, reference is sometimes made herein to the X:Drive system or to X:Drive to refer to the system as a whole. At other times, the term X:Drive indicates the user's individual X:Drive, or allocated

space. The different uses are indicated by context.

In order to effect the Shared Internet Storage Resource of the present invention, a central or distributed storage facility is provided. First and foremost is the high-speed access storage facility where files are actually stored. Such individual storage areas may be allocated in individual limited allotments, or be left open-ended and limited only by the capacity of the physical devices responsible for storage. Metadata, that is data about the files stored on the network hard drives or other storage devices, is generated and stored in a separate database. The database of metadata (the metadatabase) and the network-attached storage facility may be linked by an internal network. It is possible for the database to be stored on the same network storage facility or device on which user files are also stored. System management may select whether or not to distribute or consolidate the database with the network storage.

Also attached to the internal network is a web server that serves to generate and transmit the information to the Internet, and ultimately the user. The web server files may pass through a load balancer and/or firewall before proceeding on to the Internet. The same is similarly true for information coming into the web server from the Internet.

XML may be used in combination with JavaScript or the like to provide two means by which the Shared Internet Storage Resource of the present invention may be achieved. The first is a JavaScript object which may be transmitted to a browser program running on the user's computer. Such browsers may include ones that are well known, including Netscape® Communicator and Microsoft® Internet Explorer. Alternatively, a stand-alone application may be installed and stored upon the user's computer. This stand-alone application serves to intermediate the user commands with the web server and ultimately the metadatabase in the Internet storage device.

As an additional enhancement, the user interface may be a client program that meshes seamlessly with standard user presentations in WYSIWYG (what you see is what you get) graphic user interfaces (GUIs). As such, a drive may be shown on the user's computer and may be denominated "x:" (or "y:" or "z:", etc., depending upon user preferences). The user can then read from or write to the x:\ Shared Internet Storage Resource drive much in the same way as you would the local a:\ and c:\ hard drive.

When the user shuts down his or her computer, information that is stored on the Shared Internet Storage Resource of the present invention remains on the Internet. The user can then access such information from another computer, another geographic location, or even give permission to share files on the Shared Internet Storage Resource with others. Password protection or other security protocols may be used to limit or discriminate access to the user's files.

The Shared Internet Storage Resource of the present invention allows for direct Internet-to-Internet file transfer to a user's allocated X:Drive file space in a process referred to as "Skip the Download" or "Save to My Xdrive."

BRIEF DESCRIPTION OF DRAWINGS

Figure 1 is a schematic view of the X:Drive system of the present invention. The different tier levels are shown, along with the marking indicia of a circle, triangle, square, and star/asterisk corresponding to the same indicia in Figure 3.

Figure 2 is a schematic view of Java® library objects operating in the transactions or data exchanges occurring in the present invention.

Figure 3 is a detailed flow diagram showing the operation of the present invention. Indicia including a circle, a triangle, a square, and a star/asterisk correspond to tier levels shown in Figure 1 and indicate the level of operation of the steps shown in the flowchart of Figure 3.

Figure 4 is a flowchart showing the operation of the XDFile Enterprise JavaBean™ (EJB) used in the present invention.

Figure 5 is an overview of the Java® architecture used to effect transactions in the present invention.

Figure 6 is an alternative schematic diagram of the Java® architecture shown in Figure 5.

Figure 7 is a schematic and flowchart diagram showing the IO (input/output) for the database transactions of the present invention.

Figure 8 is a schematic diagram of the data recovery process as effected by the FileIO component of the XDFile object used in the present invention.

Figure 9 is a schematic depiction of failure recovery elements.

Figure 10 is a schematic and flowchart diagram showing virus protection effected in the present invention.

Figure 11 is a schematic and flowchart diagram of the Internet-to-resource transfer ("Skip the Download"/"Save to My Xdrive") as set forth in the present invention.

Figure 12 is a schematic and flowchart diagram of the client system used in the present invention.

Figure 13 is a Windows™ desktop display showing both the client and web-browser applications.

Figure 14 is a display of a web browser pointing to a user's X:Drive.

BRIEF DESCRIPTION OF APPENDICES

Appendix 1 is a listing of web site/server code use to achieve the present invention.

Appendix 2 is a listing of the code used on the client side to achieve the present invention in a Microsoft® Windows™ environment.

Appendix 3 is a listing of the JavaScript code used to achieve the present invention in a Sun Microsystems® Java® environment (including one on a browser).

MODE(S) FOR CARRYING OUT THE INVENTION

The detailed description set forth below in connection with the appended drawings is intended as a description of presently-preferred embodiments of the invention and is not intended to represent the only forms in which the present invention may be constructed and/or utilized. The description sets forth the functions and the sequence of steps for constructing and operating the invention in connection with the illustrated embodiments. However, it is to be understood that the same or equivalent functions and sequences may be accomplished by different embodiments that are also intended to be encompassed within the spirit and scope of the invention.

Appendices 1, 2, and 3 provide the source code for, respectively, the Web Site/Server Code of the X:Drive Shared Internet Storage Resource system of the present invention; the Windows Client Code; and the JavaScript Listings for the present invention. These Appendices are incorporated herein by this reference thereto as if set out in their entirety. It is contemplated that these Appendices provide a full, complete, and enabling disclosure to those of ordinary skill in the art or less by which the present invention may be achieved.

Additionally, the reference numbers used in conjunction with the figures are numbered such that the 100's place of the number indicates the number of the drawing figure. For example, the 600 series of reference numbers refers to Figure 6, while the 200 series refers to elements shown in Figure 2.

The present invention provides a method by which an Internet hard disk or hard drive may be achieved in a manner similar to a hard disk or hard drive available locally to the individual on the local computer. Additionally, as Internet use becomes a more familiar and everyday event for people, the resources provided by the present invention may allow the actual use of the Internet hard drive or X:Drive set forth herein to act as such a resource with the files being called up for execution for programs available and processed either locally and/or over the Internet. In light of the

foregoing, it can be seen that the present invention may act as a bridge or may pave the way towards a more inter-networked community for the use and processing of electronic information.

The virtual disk drive provided by the present invention may be selectively shared with others or kept entirely private. Additionally, and as set forth in more detail below, the use of a metadatabase provides quicker access and the ability to distribute the information regarding the legion of X:Drive accounts over a wide geographic area, enabling redundant preservation of user information by server clusters implementing Enterprise JavaBeans® (EJBs), or otherwise.

The Shared Internet Storage Resource, User Interface System, and Method set forth herein is generally referred to as "X:Drive." Context reveals whether or not the term X:Drive is referring either to the system as a whole or the individual's own account.

The X:Drive system of the present invention uses network application practices and may rely upon Java® Enterprise JavaBeans™ (EJBs) to enable distributed and clustered computing and file management environment. Along with such Java®-based and network-oriented design, the X:Drive system of the present invention also contemplates the use of open programming standards such as XML and Web-DAV (Web-based Distributed Authoring and Versioning). The use of such technology is foreseen as providing wide support by the user community as well as speed and development, refinement, and polishing.

As shown in Figure 1, the X:Drive system 100 has a multi-tiered, network-based application infrastructure. The multi-tiered nature of the system allows it to separate operations in an efficient manner. The network-based aspects of the X:Drive system allows it to disperse resources geographically as well as allow a high degree of communication between different aspects or facets of the system.

The X:Drive system may be considered enabling technology as a medium that is independent of the applications and uses to which it is applied. The X:Drive system is currently based on object-oriented principles with each application layer responsible for a discreet functionality or aspect of operation. Both hardware and software resources may then successfully experience heavy re-use with both scalability and flexibility inherently provided. While these advantageous aspects of the X:Drive system are achieved, as a multi-tiered system, X:Drive involves a higher cost of complexity and planning. Thus, those who would seek to wrongly copy the X:Drive system would do so without accruing the great expense in time and money necessary to achieve the present X:Drive system. They would ride on the backs of those who not only developed the system, but also those who got it to work right and in a commercially-reliable manner.

The use of tiers in the X:Drive system of the present invention is realized in both the network systems and the application systems involved in achieving X:Drive.

As shown in Figure 1, a variety of tiers, or layers, are present between the client 102 and the ultimate data resources 104. Between the client 102 and the data resources 104, are one or more layers or tiers, accomplishing the following.

The client 102 may be coupled to a public network 106 (such as the Internet) that may include a DNS redirector 108 as well as a load balancer 110. The public network 106 may then lead into a web server network 120. The web server may then lead into an application network 122, which in turn leads into an EJB (Enterprise JavaBeans™) network 124. The EJB network 124 may lead into a transaction network 126, which in turn leads into the data resources 104.

The client 102 may be either a web- or browser-based application or an application resident on a Windows™ X system (the X indicating the version of Windows applicable, i.e., Windows® 95, Windows® 98, Windows® 2000, etc.). Requests generally originate from the client as the X:Drive system 100 is one that operates at the command of users directing the client program. Client requests may be made versus the Hypertext Transfer Protocol (HTTP) GET/POST

function. In a preferred embodiment, the GET/POST operation may be augmented with Web-DAV extensions to the HTTP protocol. Commands are transmitted by the client 102 are sent to the DNS redirector 108, which then isolate the request via a proxy server process. A proxy server process prevents a direct connection between the client 102 and the other downstream resources in the X:Drive system 100. Such proxy serving prevents inadvertent or mischievous disruption of service by allowing only certain commands or information to be propagated through the X:Drive system 100. This prevents mischievous users from disrupting the system as such rogue commands are intercepted by the proxy server and denied further propagation.

After the client command has passed through the DNS redirector/proxy server 108, the request by the client 102 is then directed to the most appropriate facility. As the X:Drive system is scalable, facilities may be distributed geographically, even over the face of the globe. This allows, at the outset, more efficiencies to take place in the X:Drive system 100 of the present invention so that more users may be served more quickly and so that the advantageous features of the X:Drive system may be realized by the widest number of users in the quickest way possible.

Due to the construction and architecture of the X:Drive system 100, a number of machines/servers running a number of different processes may be distributed over a wide area. Broad band or high-speed access as provided by Internet backbone or the like may allow the X:Drive system to be effectively carried out over the entire face of the planet. The scalability and flexibility of the present invention augments its utility. Such advantages are further advanced by efficient use of the resources so that greater and better service can be provided.

Upon receiving the request from the client 102, the DNS redirector 108 transmits the requests on to a load balancer which may provide a second proxy process under HTTP protocol and transmit the request to the least-loaded and most-available web server on an internal, non-routable, or other server network 120.

The web server network 120 may be non-routable and may comprise a number of individual machines or servers processing the HTTP or other requests from one or more load balancers 110. Each of the web servers 140 in the network 120 may handle HTTP requests for static content, such as HTML and graphic files. The web servers may proxy all requests for dynamic content to a Java® application network 122.

As used in the X:Drive system 100 of the present invention, the Java® application networks may be non-routable. The use of non-routable facilities in the X:Drive system 100 of the present invention indicates their operation in a local area network (LAN). However, between tiers, the individual networks themselves may be available such that a web server 140 in Illinois may pass requests for dynamic content to Java® application clusters 122 in Wisconsin.

Each Java® application cluster 122 may be composed of a number of Java® application servers 142 with each server 142 handling display functions necessary for user accounts, including the generation of XML, HTML, and other instructing displays for either browser or application clients 102. If a Java® application cluster 122 receives a resource request from the web server tier 120, the Java® application cluster 122 will pass the resource request onto the Enterprise JavaBean™ EJB network tier 124.

As for the web server 120 and Java® application networks 122, the EJB network 124 may also be non-routable and operate upon a LAN. The EJB network may be an EJB cluster having a number of EJB servers 144. Each EJB cluster handles the business logic and resource access methods and protocols required for the resource requests and management. The EJB cluster (EJBC) caches memory of common resources such as the pooling of data connections and the like, as well as data objects. Resource access requests and transmissions are then passed out to the transaction network tier 126, which may also be non-routable. The transaction network tier 126 has a transaction processor 146 which controls, operates, and guarantees access and transactions on different resources. These different resources are the ultimate data resources 104 that may include NFS (Network File Server) disk arrays 150 and databases 152. The NFS

disk arrays 150 may supply the actual storage capacity for the files of generally any size. The databases 152 comprise records of information regarding each of the files (metadata) stored by the NFS disk arrays 150 under the X:Drive system 100.

By bifurcating the file information in databases 152 separate from the actual files themselves on the NFS disk arrays 150, file information and user queries can be handled much more quickly as display components of the present invention are important to provide the user information regarding the status and availability of the files stored on the X:Drive system 100. Consequently, although a user may have a hundred separate files in an X:Drive directory, he or she may be only interested in one. Consequently in order to provide the user the information necessary to make the decision as to which file to receive, move, rename, delete, or store, the use of the database provides a very quick and easy means by which such user requests can be satisfied. It is anticipated that the actual use of the file storage facilities on the NFS disk arrays 150 or the like may comprise only a part of the operations of the present invention. Having the ability to display, select, and determine file operations is one of the useful advantages provided by the X:Drive system 100 of the present invention.

Note should be taken of the non-numerical indicia present in Figure 1. Most notably, a circle is associated with the client 102, a triangle with the Java® application cluster 122, a square with the EJB network 124, and a star/asterisk with the transaction network. These non-numerical indicia correspond to those set forth in Figure 3. As different actions are performed at different tiers in the present invention, the non-numerical indicia provide an easy or visual means by which the operation of the different tiers can be indicated in Figure 3.

Figure 2 shows a logic diagram in sequence structure for the Java® library objects used in the X:Drive system 100 of the present invention. Generally, throughout the description of the X:Drive system 100 of the present invention, the prefix XD indicates "X:Drive." For example, in Figure 2 the steps/status indicators of XDError stands for X:Drive Error, and XDXML stands for X:Drive Extensible Markup Language. Likewise, the use of the term XDFile indicates X:Drive File as a Java® library object effecting and intermediating the file operations of the present invention.

In Figure 2, the Java® system 200 allows operations to be performed on the metadatabase 202 and the operating system (OS) File System 204. Additionally, the XDFile object 210 may activate or instantiate the Database.Search object 216. The XDFile object 210 may be activated, or invoked, by the FileAction object 220. The FileAction object 220 may also activate the Database.Search 216 and Database.BigSearch 222 objects. Operations of the Java® library objects in the system 200 as shown in Figure 2 may be contingent upon the SessionSecurity object 224, which may instantiate or use the Database.Search object 216 and/or the Database.Transaction object 214. The SessionSecurity object 224 may return a separate object 226 to the UserData object 230. The Database object 236 may inherit or transmit from its Transaction 214, Search 216, and/or BigSearch 222 objects.

The information generated may then be transmitted to the Database 202 for meta-information and the OS File System 204 for the actual data. If an error is generated during the operation of the Java® library object system 200, an XDError object 240 may serve to handle the error while a successful operation may be returned in the form of the XDXML object 242. In the Java® library object system 200 of Figure 2, the Database 202 may contain intelligence or programming for connection to SQL databases and the like. Options regarding the operations of the database 202 may be read from a configuration file. The Database object 236 may be able to connect multiple databases for redundancy in the case of repeated or redundantly archived information, or for functionality in order to connect to that database which responds most quickly to the requests and commands.

The Database object 236 determines which database operation to perform and/or to which database to send operations based on the type of request it receives. For example, transaction requests may demand a separate database

from those of regular query and BigSearch 222 requests. In order to maintain more efficient operation, the Database object 236 generally sends session users to the same database whenever possible so that latency and database replication is not passed on to the user.

The Database.Transaction object 214 is able to handle larger SQL statements such as those that would cause a load on the database. The Database.Transaction object 214 may spawn children classes that handle the transaction logic in order for more efficient operation.

The Database.Search object 216 is designed to handle smaller SQL statements and has children classes for specific search types, such as those along anticipated and common fields or types of information.

The Database.BigSearch object 222 handles larger, non-transactional SQL statements such as those used for reports in system accounting, monitoring, or otherwise. Children classes of the Database.BigSearch object 222 would handle specific large searches such as those that might be implemented on a monthly or other periodic basis.

The FileIO object 212 inherits and overrides Java®'s data file object. The file object contains logic to engage multiple disks or resources for redundancy and/or functionality and contains the functionalities necessary to manipulate files on the OS File System 204. The FileIO object 212 may react to the JMS (Java Messaging Service) events triggered by events on the disks of the OS File System 204.

Alternatively, one or more monitoring objects may be used to gather pertinent status information regarding the OS File System 204. When monitoring objects are used, the FileIO objects then query the common monitoring objects to determine the state of the system. In the present system, the monitoring object is denominated the Mount Point Status bean, or MPS bean, 534 (Figures 5 and 9).

Additionally, disk level transactions are carried out by the FileIO object 212. Under the management of the FileIO object 212, user accounts are able to span or traverse several disks. The spanning of such several disks enables better recovery from failure should an error occur or system resources become unavailable in an unpredictable manner. The XDFile object 210 uses FileIO 212 to handle the file system transactions. By using the Database.Transaction file object, the XDFile object 210 handles database file transactions. The XDFile object 210 coordinates transactions for both the FileIO object 212 and the Database.Transaction file object 214 to keep both synchronized and to handle failure should it occur.

The UserData object 230 holds user data for a session of the X:Drive system. A session is basically a span of time for which a user engages the X:Drive system. Methods are included in the UserData object 230 to manipulate the user status, so that the activity may be monitored, as well as whether or not the user has logged in.

The SessionSecurity object 224 uses web logic session mechanisms to create the UserData object 230. It does this by returning a separate object 226. The SessionSecurity object 224 authenticates a user's login and expires old sessions with re-direction of such old sessions to appropriate pages.

The FileAction object 220 may have children classes and contain logic for determining request types such as user requests, administration requests, etc. Tests for file action requests such as quotas and permissions, etc., may also be handled by the FileAction object 220. The FileAction object 220 accesses the file methods in the XDFile object 210.

The XDError object 240 reads a configuration file of error lists which gives each error an I.D. number. Such error lists preferably pivot on the language in which the X:Drive system 100 of the present invention is programmed. Such lists should also be able to pivot on the partner with which the X:Drive system 100 operates. Default values for the lists may be to X:Drive errors in the English language. The XDError object 240 preferably holds errors in a stack and returns any such errors from the stack. Additionally, the XDError object 240 preferably accepts new errors by code or by message.

The XDXML object 242 accepts an object and delivers as output an XML representation of a transaction or status requested by the user or client software.

Figure 3 shows the data flow through the X:Drive system 100 of the present invention, particularly that as reflected by the tiered configuration shown in Figure 1. From a starting point 300, a request is sent by HTTP POST/GET command at step 302. Web-DAV protocol may also be used and is currently considered preferable. The send request is implemented on the client 102 and is evaluated by the web server 120 as a request for static content in step 304. If the request is for static content, the file is served by the web server 120 at step 306, and the file is displayed at step 308 by the client 102.

If at step 304 the request for static content is evaluated as negative, a proxy request is issued by the web server network 120 to the Java® application cluster 122 at step 312. The request is received by the Java® application cluster (JAC) 122 and submitted to a servlet at step 314. The Java® application cluster (JAC) 122 then parses the request header at step 316. The Enterprise JavaBean™ (EJB) network 124 then authenticates the request at step 318. If authentication cannot be achieved, process control is then re-directed to the re-login page via the JAC network 122 at step 320. If authentication succeeds at step 318, the JAC network 122 then parses the multi-part form data at step 324.

The JAC network 122 then determines the type of request at step 326. The request is then submitted to the FileAction EJB 220 at step 328. The EJB network 124 then evaluates the request at step 330 in order to ensure that all the business rules and other applicable limitations are met, such as quota limitations, permissions, and the like. If the evaluation is successful at step 330, the EJB network 124 then submits the request to the XDFile EJB 210 at step 332 and on to the transaction processor 146. The appropriate actions are then taken via the transactional database 152 and the disk arrays 150. If the business rule evaluation 330 fails, an error may be generated and, as for other errors in the data flow process of Figure 3, a session error object 334 may be generated in a session error stack 336.

In effecting the data transfer to the ultimate system resources 104, evaluation is made as to the operation in step 340. If the operation is not a data read operation such as a directory listing or file read, the error stack is checked at step 342. If an error has occurred, the error status is sent to the client 102 at step 344. The client 102 then accepts the transmitted XML code and renders the appropriate display for the user at step 346. If the error stack evaluation step 342 does not reveal any error, a success message is generated at step 350, and the subsequently-generated XML is received by the client 102 and displayed by the user at step 346.

If at the evaluation step 340, the operation is not a data read action, the error stack is checked at step 352 much in the same way as it was at step 342. If an error has occurred, the error status is sent to the client 102 at step 354. The error status message is then received as XML code by the client 102 at step 346 and displayed to the user. If at evaluation step 352 the error stack reveals no errors, the evaluation is then made by the EJB cluster as to whether or not the operation is a file read at step 360. If the operation is a file read, the data stream is converted to a network stream and transmitted as a file to the client 102 by the Java® application network 122 at step 362. The data is then accepted by the client 102 and served to the user at step 364.

If at evaluation step 360 the operation is not a file read (see Figure 4), then by elimination, the action is a request for file metadata such as a directory listing indication of file attributes or the like. At step 366, the metadata retrieved from the database 152 is then translated into XML format by the EJB cluster 124. The XML data is then transmitted to the JAC network 122, which encapsulates the XML from the network and sends it on to the client at step 368. The JAC network 122 then sends the encapsulated XML to the client 102 for rendering and display at step 346.

As indicated in the description above with regards to Figure 3, users utilizing the client system 102 to connect to the X:Drive system 100 do so via the public Internet and then submit requests and receive replies effecting or indicating

the user's requests. Requests for file manipulations, such as uploads, downloads, copies, moves and updates travel through each functional layer of the X:Drive system 100.

The core of the EJB cluster, and as indicated in Figure 2, the XDFFile EJB provides core effectiveness in the present X:Drive system 100. The XDFFile EJB 210 is a multi-tiered component. The X:Drive system 100 stores file metadata (such as directory structure, file name, file attributes, etc.) in the database 152 for fast retrieval, sorting, searching, linking, and other capabilities beyond standard file systems. The actual file data is stored by the X:Drive system 100 in network-attached storage units or storage area networks such as those shown in Figure 1, the NFS disk arrays 150.

To access files that exist in this hybrid environment (bifurcated between file information and file data), X:Drive uses the XDFFile object 210 to manipulate both files and file data in two-phase committal transactions. Figure 4 shows the details of these transactions.

In Figure 4, the XDFFile EJB system 400 allows entry at any one of the five darkened triangles. If the action is to be a copy, entry is made at the copy entry point 402. If the action is a file read, entry is made at the file read point 404. If the action is a file write, entry is made at the file write point 406. If the action is a file delete, entry is made at the delete point 408. If the action is a file move, entry into the XDFFile EJB 210 is at the move entry point 410.

Beginning first with a file copy action beginning at the copy point 402, the evaluation of the operation occurs at step 420, where determination is made whether or not the action is a read transaction. If the action is a read transaction, program flow proceeds onto the read action and entry point 404. The corresponding database action 424 is then taken. As the action is a read transaction, the corresponding database record is read and evaluation is made as to whether or not the database action, in this case read action, has been successful at step 428. If the read action is not successful, the changes are then rolled back, if any, at step 432. An error is then returned at step 436 and the XDFFile object awaits further instructions. If the evaluation at step 428 regarding the database action was successful, action can then be taken on the actual file itself on the OS File System 204 at step 440. In the present case, the FileOS Action 440 is a read action, and the file may be read into a temporary buffer or other memory space. The FileOS Action is evaluated for success at step 444. If the FileOS Action step 440 was unsuccessful, a fatal error is returned at step 448, and the changes, if any, are rolled back at step 452. If the evaluation at step 444 was successful, evaluation is made as to whether or not the action was a copy read at step 456. If the action was a copy read, return is made to the copy entry point 402 at step 464 in order to perform the write portion of the copy function. If the evaluation at step 456 indicates that the action was not a copy read action, evaluation is made at step 468 to determine if the action was a move/copy action. If the action was a move/copy action, control is then directed towards the move entry point 410 via step 472 in order to delete the original file as the success of the move/copy transaction at evaluation step 444 indicates the success of the file write step of the FileOS Action step 440. Program control is then turned over to the move/action entry point 410 so that the original file may be deleted at its original location via the delete entry point 408.

If the move/copy evaluation step 468 indicates that not only was the action not a copy read, it was also not a move/copy, then the action is committed to the system at the ultimate system resource level 104 at step 480 and an indication of success is then returned at step 484.

Upon reaching the move entry point at 410, evaluation is made at step 490 to determine whether or not the transaction is a copy transaction. If it is a copy transaction, the program then enters and executes the copy entry point 402. If not, the delete entry point 408 is activated to effect the remainder of the move transaction.

Consequently, it can be seen that a variety of actions take place depending upon the state of the XDFFile EJB 210 at the database action 424 and FileOS action 440 steps.

In performing file reads and writes, simple one-step actions are taken because neither of these read or write actions are either copy reads 456 or move/copy 468 and so they fall into the system commit 480 and return a successful indication at step 484. The same is generally true for the one-step delete action. Consequently, whenever a user wants to read, write or delete a file, entry can be made into the respective entry points at 404, 406, and 408. Errors are returned when necessary.

However, the copy action 402 and the move action 410 require multiple loops through the XDFFile EJB 210 in order to effect their operations. For the copy function 402, the initial read must be made successfully with the evaluation step 456 then prompting the write step to occur by the return to the copy entry point at step 464. The read transaction step 420 is then evaluated in the negative and the write entry point/action 406 is invoked with the database action occurring at step 424 to write the new information to the transactional database 152 and, if successful, the FileOS write action for the data at step 440. If the file write is successful, the evaluation at step 456 as to whether or not the action is a copy read is answered in the negative as is the evaluation of the transaction as to whether or not is a copy transaction executed under the move action at step 468. The resources are then committed, temporary resources are released, and the success indication is returned at step 484.

Consequently, for a copy transaction 402, the loop is first made through the read function 404 and then the write function 406. For the move action at entry point 410, a copy transaction is first executed with the two-loop operation as set forth previously. Upon completion of the copy action, the delete action 408 is implemented in order to erase the original file and its file data. Upon the third loop through the delete step 408, the transaction is neither a read under the copy command at step 456 nor a copy under the move command at step 468. Consequently, the move function has successfully completed, the system resources are committed at step 480, and a success indicator is returned at step 484.

In Figure 5, an overview of the Java® architecture of the X:Drive system 100 of the present invention is shown. The Java® architecture 500 shown in Figure 5 may generally arise from the client 102. A file action container 504 has certain attributes and operations as do the other beans of the architecture 500. Contained within the file action container 504 are a number of stateful, stateless, and entity beans, as well as other containers having other beans. The file action container 504 contains two stateful beans: a user data stateful bean 506 and a process request stateful bean 508. The user data stateful bean 506 has a user info entity bean 510 and a security stateless bean 512.

The process request stateful bean 508 contains a single container, the XDFFile container 520. The XDFFile container 520 contains three (3) beans and a container. The three beans of the XDFFile container 520 are: a database IO stateful bean 522, a file IO stateful bean 524, and an admin stateful bean 526. The container is a recovery container 530 which contains a recovery IO stateful bean 532, a mount status stateful bean 534, a recovery admin stateful bean 536, and a recovery process stateful bean 538:

As indicated by the nature of the beans carried by the containers, stateful beans generally carry information about the state of the bean, process, or otherwise as useful information for the ends and operations of the X:Drive system 100 of the present invention. Stateless beans generally carry no state information, and entity beans are generally for information or identification only. As Java® beans are objects intended to carry both data and processes in association with one another, it is up to the operations of the X:Drive system 100 of the present invention to selectively and appropriately activate the beans and enable the proper actions to take place. The file action container 504 is shown in alternative representation in Figure 6. In Figure 6, a client 102 issues a user authentication request 602 and an operation request 604. The user authentication request 602 is passed into the user data stateful bean 506 in the file action container 504. The operation request 604 is passed into the process request stateful bean 508. The user information entity bean 510 then transmits information to a user information database 610, as does the security stateless bean 512. The process

request stateful bean uses a first property file 612 that is loaded upon deployment of the XDFFile container 520. The property file is loaded into the admin stateful bean 526 for use with the OS file system 204. A Java® transaction server 620 may operate in conjunction with the database 152 as well as the OS file system 204 in order to process the operation request 604. The second property file 630 may be loaded by the recovery admin stateful bean 536 upon the bean's deployment. The recovery IO stateful bean 532 and the recovery admin stateful bean 536 both transmit information to the recovery queue storage buffer 640. The mount status bean 534 operates in conjunction with the mount status of the system 650.

The recovery container 530 is called when once a failed resource begins to recover. Further description of the recovery process is given below. However, Figures 5 and 6 operate in tandem to show linearly (Figure 5) and organically (Figure 6) the structure and operation of the XDFFile object 210.

Figure 7 shows the detail of the XDFFile database component. A transaction processor (such as Tuxedo from BEA) works in conjunction with the database transaction object 214 as well as the FileIO object 212 to provide a robust and reliable system. Both the database transaction 214 and the FileIO 212 objects include logic and/or programming to handle situations where database or disk array access cannot be guaranteed. The database.transaction object 214 handles the inherent doubt present in the system by using replicated or repeated clusters of databases. The replication process creates latency or delay, in the system. In order to accommodate this latency, the database transaction object 214 uses a session object (a data construct representing a user session on the X:Drive system 100) to determine if the user's request can be transferred, or replicated, from one database cluster to another, in case of future system failure.

An important aspect with respect to the reliable operation of the X:Drive system 100 is the need to separate databases into functional groups. While the query database may be optimized for quick and small queries and while a transaction database might be optimized for fewer, larger, more time consuming updates, the database layer 236 in the X:Drive system 100 allows for associating SQL commands with different database clusters based on functionality. Additionally, the X:Drive database layer 236 is configured for consolidation and addition of databases on the fly.

As shown in Figure 7, the SQL command 710 is issued and passed to a SQL command evaluator 712. A SQL evaluator determines the SQL type so that the SQL can be sent to the appropriate database type (that is, in the X:Drive system 100, the transaction database 150, the query database 152, or both).

Upon determining the database type of the SQL statement 712, the database preference is evaluated at step 714 to determine if the user should be sent back to the same database. If the user is not to be sent back to the same database, the database currently bearing the least load is found in step 716, and query is then made in step 718 to ensure that the selected least-loaded database is still up, running, and available. If it is, a specification regarding the pooling of database resources is created 720 and transmitted to the database object 236. Database object 236 then takes the SQL command and passes it to the appropriate database, either the transaction database 150 or the query database 152 via associated connecting pools 730.

If at step 718 the least loaded database is not available, an alternative database must be used and query is made at step 736 to determine whether or not the alternate database is up. If the alternate database is not up and the evaluation step 736 fails, additional databases may be queried or, as indicated in Figure 7, a fatal error may be generated at step 738. If the alternate database is up, a pool specification 720 is generated and passed to the database object so that the SQL command may be implemented upon the transactional 152 databases via the connection pools 730.

If at step 714 the user must be sent back to the same database, query is made at step 740 to determine if that database is still up. If it is, the request is passed to the pool specification 720 where it is subsequently passed to the database object 236, on to the connection pool 730, and the appropriate database, either the transaction database 150 or

the query database 152. If the same database is not up and the evaluation at step 740 fails, an alternative database must be used, but the SQL request is queried at step 744 to determine if the SQL command is transferable to the alternate database. If not, a fatal error occurs at step 746. If the SQL command is transferable, query is made at step 750 to see if the alternate database is up and active. Should the evaluation fail, subsequent databases may also be queried if the SQL command is transferable. However, as shown in Figure 7, if the second database is unavailable, a fatal error may be generated at 746. Otherwise, the database is up, and the evaluation at step at 750 is successful and the command is made available to the database object 236 via the pool specification standard 720 and on to the databases through the connection pools 730.

In order to ensure proper operation of the XDFile database object 210, a database status monitor 760 persistently and on-goingly queries the databases 150, 152. The status is then returned to a database status object 762. the database status object may provide information to the recovery container 530 of the XDFile object 210.

The recovery mechanism for the X:Drive system 100 of the present invention is shown in Figure 8. The FileIO object 212 uses a recovery object such as the recovery container 530 to handle write transactions 406 (as opposed to read transactions 404) when the transaction processor 214 fails. The recovery object is transparent to the user, making it easier and more convenient for the user to use the X:Drive system 100 while decreasing the concern that such a user would have in case of a power outage or other failure in one part of the X:Drive system 100.

The FileIO object 212 reports an error to the user, but informs the user that her request was stored in the X:Drive system 100 and that the X:Drive system 100 will try to apply the change as soon as possible. If the storage unit, represented as a mounting point in the EJB cluster becomes unavailable for write transactions 406, the monitoring client 760 updates the EJB network 124 that the status of the mounting point is "down." Once the mounting point is available and checked for data integrity, the status is updated from "down" to "recovery" and the recovery object 530 is called to apply all queued requests for the file action container 504. This keeps the user from catastrophically losing uploads and other file writes, but may cause some delay in file reads.

In the recovery system 800 of the present invention, the multi-connected pooled database object, the recovery-enabled FileIO object 212, and the transaction processor 146 work together to create a resource layer offering high availability, recovery, and scalability. Additionally, this resource layer (encapsulated in the XDFile EJB 210) lends itself to replication of the data, both geographically and locally. Such replication preferably has the three essential traits of being off-site, application-driven, and accessible. With this level of controlled replication, secondary X:Drive clusters are enabled in geographically diverse locations in order to enhance the reliability of the X:Drive system 100. Consequently, data loss from one data center or even the physical loss of an entire data center would not cause loss of customer data or access. Re-direction would occur dynamically and this information would be replicated in a plurality of sites across the X:Drive system 100, the query or metadata databases provide multiple pointers to the user's data.

In the recovery system 800 of Figure 8, the recovery system is initially initiated when the MPS Bean 534 is set for a mode to detect mount point recovery at step 804. At step 804, a recover method is called and the external mount point is checked. Query is made at step 806 to evaluate whether or not recovery is already occurring. If recovery is already occurring, an exception is thrown at step 808 and exit is made at this finish point. If recovery is not already occurring, a list of mount points in recovery mode is generated in step 810. Additionally, at step 812 a list of mount points which are down is also generated. Query is made at the evaluation step 818 as to the presence of available recovery objects in the recovery queue. If no such objects are available in the queue, the disk or other database is set into the "up" mode at step 820. The queue for that disk is then unlocked in step 822, and the recovery process is complete at step 824. If at evaluation step 818 recovery objects are still in the queue, evaluation is made as to whether or not the system has gone

past the lock count at step 830. If so, the queue for the disk in recovery is locked at step 832 for both the lock count evaluation 830 and the queue lock 832 step, control is then directed to the evaluation step as to whether or not the target file exists 834. If the target file does not exist and the evaluation at step 834 fails, the recovery object is removed from the queue at step 840. The status of the recovery is subsequently put in the request for alert queue at step 842 and return is then made to the query step 818 to determine whether or not objects are still available for recovery in the queue.

If the target file does exist when evaluated at step 834, evaluation is made as to whether or not the request is more current than the file at step 850. If the request is older than the current file, the recovery object is removed from the queue at step 840, and the status for the request is put in the request or alert queue 842 and control returns back to the evaluation step 818 to see if any further recovery objects are available in the recovery queue.

If, in evaluating the request, it is found that the request is more current than the file, the request is submitted to the XDFFile object 210 at step 852. The submission of the request to the XDFFile object 210 is not recoverable. If the submitted request is successful as indicated by the evaluation at step 854, the recovery object is removed from the queue at step 840, its status is put into the request for alert queue at step 842 and evaluation is made at step 818 as to the presence of any additional recovery objects in the recovery queue. However, if in submitting the request to the XDFFile object 210 at step 852 the submission fails, query is made at step 860 as to whether or not the mount point has gone down. If at step 860 the mount point is still up, the request from this mount point is ignored at step 862 and the queue for the disk is unlocked at step 864. Control of the program is then returned to the recovery object availability query in evaluation step 818.

As shown in Figure 9, the mount point status bean 534 has UP, DOWN, and RECOVERY states. This bean is applicable to the file database 150, as well as user disks 970, 972 as well as recovery disks 974, 976. Additionally, the recovery admin stateful bean 536 is directed towards the recovery database 980 in order to effect the recovery process 800.

In order to effect virus scanning and repair features, the X:Drive system 100 preferably uses the Java® JNI (Java Native Interface) to access a Norton Anti-Virus or other dynamically linked library (NAV.DLL) to scan files for viruses via a Java® servlet. The Java® servlet runs on a Windows™ version X server and can use JNI to make calls to the NAV.DLL dynamically linked libraries. In effect, the Windows™ X machine becomes a specialized NAV.DLL server located at the EJB network layer 124 of the X:Drive system 100, on a sub-network of the resource network. The logic integrating the NAV.DLL dynamic linked libraries with all X:Drive file writes is shown schematically in the flow diagram in Figure 10.

As shown in Figure 10, the virus scanning sub-system 1000 takes the file/transaction ID 1002 and a transaction ID 1004 from a user 1006. The file/transaction ID 1002 is passed to a file write process 1008 executed by a SUN® or other web server 1010. The file is written to both the database generically indicated at reference 1020 and to a temporary file storage area 1022. The file write process 1008 passes the file transaction ID to the Norton Anti-Virus (NAV) process 1024. Within the NAV process 1024 is NAV scanner 1026. The NAV scanner monitors the data stream or otherwise to determine and detect the presence of any viruses. If upon evaluation the NAV process 1024 detects a virus at evaluation step 1028, data sink action is taken with respect to the database 1020. If no virus is detected, the sequence moves to its final termination at step 1030 and data sink action is taken with respect to a temporary file on medium 1032.

While both the file and transaction ID 1002 are delivered to the file write process 1008, the transaction ID alone 1004 is transmitted to a fetch location info step 1040 on a SUN® or other web server 1010. The fetch location info step 1040 transmits its results to an evaluation step 1042, which determines whether or not the file is in the temporary storage area 1022. If the file is in the temporary area, the file's upload status is shown in step 1044. If the file is not in the

temporary medium 1022, virus information is fetched at step 1050 in the file status process 1036.

Once the virus information has been fetched, it is evaluated as to whether or not there is a virus present at step 1052. If there is no virus detected, then the virus evaluation terminates and a display of same may be made at step 1054.

However, if evaluation step 1052 indicates the presence of one or more viruses, a plurality of virus options may be shown and presented to the user at step 1060. Among the virus options available are: the cleaning of the virus at step 1062, moving the virus to a different location at step 1064, and/or deleting the virus in step 1066. If step 1064 is taken with the move of the virus-laden file despite its infectious nature is made, movement of the file with its final destination is made in step 1070.

As shown in Figure 10, a number of data sink actions are taken with respect to information. Additionally, as indicated by Figure 10, the NAV process 1024 is a separate entity and may be considered to be a JAVA® servlet/daemon living on specialized Windows® NT or other servers.

In order to make resources available on an on-going basis to the virus scanning sub-system 1000 of the present invention, a chron file 1074 (a file executing commands on a periodic basis according to the time) is used to remove old files from a first temporary storage resource 1002.

Figure 11 shows the Skip the Download/Save to My Xdrive system where a file on the Internet can be transferred over to an individual's X:Drive at generally data speeds far faster than those available to the end user. This allows the user to exercise dominion and control over the file without having to bear the burden of downloading it to the local computer at the present moment. Once the transfer has taken place across the Internet from the host to the X:Drive system 100, then the user may download the file stored in his X:Drive directory to his local computer at his convenience.

As X:Drive exists on the Internet network, transferring a file from one network resource (such as a web or FTP server) to the user's X:Drive is made much faster from the user's standpoint by by-passing the local connection to the user and allowing the user to submit the transfer request directly to the X:Drive network for execution. The X:Drive system 100 then downloads the requested data from the target server to the user's X:Drive over the presumably higher speed connections of the public Internet.

As shown in Figure 11, the Save to My Xdrive system 1100 first has the user 1110 submit the URL at step 1112. In order to access the X:Drive system 100 of the present invention, the user submits the URL as well as his or her user name and password at step 1114. Upon submitting the URL and the appropriate verification information, evaluation is made of the information for authentication purposes at step 1116. If the evaluation fails and authentication is not achieved, a login form is displayed in conjunction with the previously-indicated URL at step 1118. If the request is authenticated, it is submitted to the STD/STMX (Skip the Download/Save to My Xdrive) queue 1132 at step 1130. A status process is then spawned at step 1134.

Save to My Xdrive status is then checked on an on-going basis by using the queue in the temporary storage area at step 1136. Query is made as to whether or not the transfer is complete at step 1140. If the transfer is complete at step 1140, then the successful completion is indicated to the user at step 1142. However, if the transfer is not complete, query is made as to the presence of any transfer errors at step 1146. If an error has occurred, an error message is displayed to the user at step 1148. However, if the transfer is incomplete but no errors have occurred, the same is then displayed to the user at step 1150, and a short pause is taken at step 1152 for re-invoking the check STD process at step 1136.

Once the STD queue 1132 receives the request, a daemon process processes the request from the STD queue at step 1160. Query is made as to the business logic of the queued request at step 1162. If the request fails the business logic check 1162, the status is updated at step 1164. Control may transfer back to the STD queue 1132.

If the business logic check succeeds at step 1162, the URL site is contacted by the X:Drive system 100 at step

1170 and the download process is activated. The data transmitted by the URL is then saved in temporary X:Drive space in step 1172, with the data being transferred then to the user data space at step 1174. The URL site 1180 may exist anywhere on the Internet so long as it is available to the X:Drive system 100. In a similar manner, a temporary storage space 1182 may also exist anywhere on the Internet so long as it is accessible and controllable by the X:Drive system 100.

Upon transferring data to the user's data space as shown in step 1174, query is made as to the success of the transfer at step 1188. For either success or failure of the successful file transfer at evaluation step 1188, the status is updated at step 1164 and is passed on to the STD queue 1132 until either success or an error is finally achieved. The status process spawned at step 1130 monitors the update status generated by step 1164 and displays the status to the user during and after the download of the file from the Internet to the user's X:Drive system.

Figure 12 shows a schematic and flowchart diagram for the client system generally used under Microsoft® Windows™ for achieving the present invention. The X:Drive system offers its clients two basic services: a file access service by which files can be uploaded and downloaded to and from X:Drive, as well as a file manipulation service from which file metadata can be obtained and manipulated. Both of these services rely upon the context of their usage. For example, the web client of the present invention uses native upload and download features as well as dialogs in the user's web browsers to facilitate the service.

With the use of the web browsers on the local machine, Windows® X clients use the Windows™ TCP/IP stacks inherently present with the Windows® version X operating system. All the file transfers effected by the X:Drive system can take place as HTTP POST/GET or, preferably, Web-DAV transfers. Generally, two basic layers are present in the file manipulation servers of the X:Drive system 100 of the present invention. An XML parser operates in conjunction with an XML data displayer. By coordinating the two basic layers of the file manipulation service, the server is able to respond with generally the same XML code to all clients. The client is then responsible for converting the XML to a relevant data structure and displaying the XML in an appropriate context. In the present invention, the JavaScript web client receives the XML code and parses it into a JavaScript data structure. A display layer in association with the client and/or browser renders the data structure as an HTML document. The Windows® X client parses the same XML code, but the display layer renders the data structure into a device listing that is understood by the Windows® version X operating system. The importance of this layered architecture is that it generally makes trivial the creation of new clients. Instead of simply creating dynamic web pages (and thus limiting service to web browsers alone), the X:Drive system 100 can enable many platforms, such as operating systems, without altering the server structure. Most platforms come with some sort of XML parsing layers, and many platforms come with display layers ready made. Consequently, the time to market may generally be considered low and efficient establishment and implementation of the X:Drive system 100 of the present invention can be achieved fairly quickly. Additionally, expansion into new platforms generally becomes much quicker as no alteration of the server structure generally needs to occur as Java® and related program functionalities are highly portable from one system to another.

In the client system 1200, as shown in Figure 12, the client 102 has a file access service 1202, including a request processing layer 1204 coupled to a network I/O layer 1206. Commands and data are then transmitted to the server side of the X:Drive system 100 where the server side request processing layer 1210 transmits the data to a query evaluating whether or not the request is one for metadata at step 1212. If the evaluation fails and the request is not one for metadata, the network I/O layer 1216 and the resource access layer 1218 are invoked in order to provide access to and operation of the transaction database 152.

If the request for metadata query at step 1212 succeeds, the request is passed on to the resource access layer 1218

and on to the XML generation layer 1220. The response to the request from the metadatabase 150 is transmitted to the file manipulation service system 1230 of the client 120. The XML transmitted by the XML generation layer 1220 is received by the file manipulation service 1230 as well as its XML handler 1232. The XML is then passed on to the XML parser layer at step 1234 to arrive at a data structure 1236 that is then ready for display and so is passed on to the data display layer 1238 for display to the user who may then re-initiate the process by implementing the file access service 1202.

Figure 13 shows the X:Drive system 100 as implemented on a Windows™ X machine, in this case, a Windows '98 machine (an Intel-based personal computer running the Microsoft Windows '98 operating system).

The second frontmost window 1310 of Figure 13 is headed by the inscription "My Computer" and shows the presence of a drive at logical letter X: 1312 with the X:Drive logo and the label www.xdrive.com (X:). This is an example of the user interface provided by the client application. The X:Drive system is transparent to the user and functions as any other drive present on the system.

If the user were to click on or activate the X:\ drive on the My Computer window 1310, the second window 1320 appears (partially obscuring the "My Computer" window 1310) and shows the listing under the X:\ Drive. The address of the window 1320 shows the location of the directory as being at X:\ 1322.

Also shown in Figure 13 is the desktop icon 1330, the start menu icon 1336, and the system tray icon 1340. These icons accompany the client program 102 and provide greater functionality for the user. Each icon serves to activate the client program in accordance with user-settable preferences.

Figure 13 also shows the web-based application 1350 in the background, behind the My Computer 1310 and X:\ 1320 windows. The web-based application window 1350 is shown in Figure 14. Note should be taken of the exact correspondence between the directory structures of web-based application window 1350 and the client-based application window 1320. This correspondence provides the user with a uniform, familiar, and dependable interface upon which the user can rely.

As set forth above, the three accompanying Appendices are incorporated herein in their entirety, as is the previously filed provisional application.

While the present invention has been described with regards to particular embodiments, it is recognized that additional variations of the present invention may be devised without departing from the inventive concept.

INDUSTRIAL APPLICABILITY

It is an object of the present invention to provide a Shared Internet Storage Resource on which users may store and retrieve files to make them available to themselves, or possibly others, throughout the Internet.

It is an additional object of the present invention to provide all manner of file access and control generally available to files local to the users for such Internet-stored files.

It is an additional object of the present invention to provide an easy-to-use and readily understood user interface through which files may be stored, retrieved, and manipulated on the Internet.

It is an additional object of the present invention to gather metadata regarding such files and to store such metadata in a database.

It is yet another object of the present invention to provide a plurality of means by which Internet-stored files may be manipulated and controlled.

It is yet another object of the present invention to provide a browser-based access to Internet-stored files.

It is yet another object of the present invention to provide stand-alone application access to Internet-stored files.

It is yet another object of the present invention to provide means by which Internet files may be stored on an Internet resource by a direct Internet-to-Internet transfer subject to the control of a remote or limited-resource user.

These and other objects, advantages, and the industrial utility of the present invention will be apparent from a review of the accompanying specification and drawings.

Web Site/Server Code

###addspace.cgi	1
###client_info.cgi	9
###cookie.cgi	12
###download_client.cgi	19
###email_change.cgi	21
###error.cgi	23
###explorer.cgi	24
###explorer_user_data.cgi	28
###file_load.cgi	30
###file_save.cgi	34
#! file_upload_stat.cgi	36
###folder_create.cgi	42
###forgot_password.cgi	44
###forgot_username.cgi	47
###frame_generic.cgi	50
###get_a_shared_file.cgi	52
###get_a_shared_file_download.cgi	54
###login.cgi	57
###logout.cgi	65
###navbar.cgi	66
###password_change.cgi	68
###promo.cgi	71
###removespace.cgi	74
###selected_delete.cgi	79
###selected_rename.cgi	80
###settings_save.cgi	82
###share_a_file.cgi	85
###signup_account.cgi	91
###signup_form.cgi	105
###signup_success.cgi	111
###signup_toc.cgi	113
###skip_the_download.cgi	115
###skip_the_download_status.cgi	122
###tell_a_friend.cgi	131
###web_unauthorized.cgi	135

###addspace.cgi

```
#!/usr/bin/perl
#####3
##  addspace.cgi - processes additional space requests using Epoch's
##  do_approval library function
##  written by Karen Eppinger
#####3

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Error;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::Deal;
use XDrive::DatabaseO::Table::Item;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::Client::Actions;
use XDrive::Client::Quota;
use XDrive::Sale::Purchase;
use Mail::Sendmail;
use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Template;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::CGI::Cookie;
use EpochClient_ssl;

use strict;

$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)); # Make %ENV safer

&main();

#####
## main: main function calls all others
##
##
#####

sub main
{
    ##the hash that will be filled in and send to the Epoch function
    my %hData;

    my $oCGI = CGI->new();
    my $oErrors = new XDrive::Error;
    my $oDBH = XDrive::DatabaseO->new();

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    #####
    ## Validate the user and if an error happens during
    ## the validation process die redirect to the error cgi
    #####
    my $oToken = xd_security_check($oDBH,$oCGI,$oErrors);

    if ($oErrors->Occurud)
```

```

    {
        xd_fatal_error($oCGI,$oErrors);
    }

    $hData('ipaddr') = $oCGI->remote_addr();

    if ($hData('ipaddr')=~/^192.168.2/)
    {
        $hData('ipaddr')='0.0.0.0';
    }

    my $sUserName = $oToken->data('user');
    # my $sPartnerCode = $oToken->data('partner_code');
    my $sPartnerCode = $oCookie->getElement('partner');
    my $oTemplate = new XDrive::Template
        (
            {
                'partner_code' => $sPartnerCode
            }
        );

    ##used to figure whether to give user the form or process the form
    my $sAction = $oCGI->param("action");

    ## if the action is a request type, we give the user the form
    if ($sAction eq 'process')
    {
        ##get the date from the form already pre-screened by javascript
        my $returnValue = GetFormData(\%hData,$sUserName,$oCGI,$oDBH);
        if ($returnValue)
        {
            ##call the Epoch function that processes the transaction
            my $sReturnCode = do_approval(%hData);

            ##if we've been approved $return will contain a number that
            is
            ##7 characters and starts with a Y followed by 7 digits
            ##only change user's quota if approved
            ##else let them know there was a problem; all problems
            start with N
            ##return code could be logged in our database to track
            tranactions

            ##truncate expressions longer than 32 characters
            if (length($sReturnCode)>32)
            {
                $sReturnCode = substr($sReturnCode,0,32);
            }

            if ($sReturnCode=~m/^Y/)
            {
                ##if transaction went through, give them more space
                ##and show them the ok screen
                my $error =
                &WriteToPurchaseDatabase($sReturnCode,\%hData,$sUserName,$oDBH);
                if ($error)
                {
                    &TransactionOK($sReturnCode,
                    \%hData,$sUserName,$oTemplate,$oDBH,$oToken,$oCGI,$oErrors);
                    $oDBH->commit();
                }
                else
                {
                    ##error inserting into the database
                    &TransactionBad('141',$oTemplate,$oErrors);
                }
            }
        }
    }

```

```

        $oDBH->rollback();
    }
    elsif ($sReturnCode=~m/^N/)
    {
        ##tell them there was a problem
        ##for some reason we get this returned with
        $sReturnCode=~s/~//;
        my $error =
&WriteToFailedDatabase($sReturnCode,\%hData,$sUserName,$oDBH);
        &TransactionBad($sReturnCode,$oTemplate,$oErrors);
        $oDBH->commit();
    }
    else
    {
        ##There was a problem connecting to server
        my $error =
&WriteToFailedDatabase('COULDNOTCONNECT\n',\%hData,$sUserName,$oDBH);

        &TransactionBad('COULDNOTCONNECT\n',$oTemplate,$oErrors);
        $oDBH->commit();
    }
    else
    {
        ##this is someone trying to use the
        ##bogus card numbers and isn't one of us
        ##don't bother writing to database because
        ##it is caught before going to Epoch
        &TransactionBad('NMYBADCARD\n',$oTemplate,$oErrors);
    }
    $oDBH->disconnect();
}
elsif ($sAction eq 'intro')
{
    &ShowIntroPage($oTemplate,$sPartnerCode,$sUserName,$oToken,$oCGI,$oError
rs);
}
else
{
    &ShowForm($oTemplate,$sUserName,$oErrors);
}
exit;
}

```

```

#####
## GetFormData:  Fills in the hash that is required by Epoch's function
## Fill in one field at a time because not all fields on the page should go
## into hash plus a few fields don't come from form
#####

```

```

sub GetFormData(\%,$,,$)

```

```

{
    my $hData = shift;
    my $sUserName =shift;
    my $oCGI = shift;
    my $oDBH = shift;

    my $value = 1;

    ##these are mandatory to process the tranaction
    ##javascript checks insure users fill the fields with the proper data

```

```

    $hData->{'transtype'}='approve';
    $hData->{'co_code'}='xdr';
    $hData->{'pi_code'}= $oCGI->param("pi_code");
    $hData->{'cardnum'}= $oCGI->param("cardnum");
    $hData->{'cardexp'}=$oCGI->param("cardexp");

    ##someone is trying to access from a site other than ours and use the
    free credit card
    if (($hData->{'cardnum'} eq '4121371122223333') || ($hData->{'cardnum'}
    eq '4111111111111114'))
    {
        if ($hData->{'ipaddr'} ne '0.0.0.0')
        {
            $value=0;
        }
    }

    ##not required but used to check for fraud
    $hData->{'cardname'}= $oCGI->param("cardname");
    $hData->{'street'}=$oCGI->param("address");
    $hData->{'city'}=$oCGI->param("city");
    $hData->{'state'}=$oCGI->param("state");
    $hData->{'zip'}=$oCGI->param("zip");
    $hData->{'phone'}=$oCGI->param("phone");

    ##get email out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, $oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);
    my $sUserSeq = $oDiskAccount->fetchColumn('USER_SEQ');
    $oDiskAccount->finish();

    my $oUserInfo = XDrive::DatabaseO::Table::UserData->new(undef, $oDBH);
    $oUserInfo->loadWhere('SEQ', $sUserSeq);
    $hData->{'email'}=$oUserInfo->fetchColumn('EMAIL_ADDRESS');
    $oUserInfo->finish();

    return $value;
}

#####
## ShowIntroPage: called to show the intro page
##
#####

sub ShowIntroPage($,$,$)
{
    my $oTemplate = shift;
    my $sPartnerCode = shift;
    my $sUserName = shift;
    my $oToken = shift;
    my $oCGI = shift;
    my $oErr = shift;

    my ($nUserSeq, $oUserData);

    my $oAction = new XDrive::Client::Actions($oToken, $oCGI);
    my $quotaAvailable = $oAction->QuotaFree();
    $quotaAvailable = sprintf("%.2f", $quotaAvailable/1024);

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, undef);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

```



```

my $nUserSeq = $oDiskAccount->fetchColumn("USER_SEQ");

my $oSearch = XDrive::DatabaseO::Search->new($oDiskAccount-
>fetchDBO());
my $items = $oSearch->XDGetItemsForSale($nUserSeq);

my $itemString='';
my $i;
for $i(0..$#{ $items })
{
    ##now using the code, get the description for the item in the
    ##proper language. This is kept in List.pm
    my $code = "EPOCH_$items->[$i][1]";
    my $description = $oErr->ReturnMessageGivenCode($code);
    $itemString .= "<LI>$description";
}

## Load the required template HTML files.
$oTemplate->load('addspace_intro.thtml');
$oTemplate->tags
(
    (
        'products' => $itemString,
        'quota' => $quotaAvailable
    )
);
$oTemplate->clear;

print "Content-type: text/html\n\n";
print $oTemplate->get();
}

```

```

#####
## ShowForm: called to show the user the blank form
##
#####

```

```

sub ShowForm($,$)
{
    my $oTemplate = shift;
    my $sUserName = shift;
    my $oErr = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,undef);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);
    my $nUserSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oSearch = XDrive::DatabaseO::Search->new($oDiskAccount-
>fetchDBO());
    my $items = $oSearch->XDGetItemsForSale($nUserSeq);
    my $oDeal = XDrive::DatabaseO::Table::Deal->new(undef, $oDiskAccount-
>fetchDBO());

    my $itemString='';
    my $i;
    for $i(0..$#{ $items })
    {
        $oDeal->loadWhere("ITEM_SEQ", $items->[$i][0]);
        my $pi_code = $oDeal->fetchColumn("PRODUCT_CODE");
        my $code = "EPOCH_$items->[$i][1]";
        my $description = $oErr->ReturnMessageGivenCode($code);
    }
}

```

```

        if ($i == 0)
        {
            $itemString .= '<input type="radio" name="pi_code" value="" .
$pi_code . '" CHECKED>' . $description . '<BR>';
        }
        else
        {
            $itemString .= '<input type="radio" name="pi_code" value="" .
$pi_code . '">' . $description . '<BR>';
        }
    }

```

```
$oDeal->disconnect();
```

```

## Load the required template HTML files.
$oTemplate->load('addspace_request.shtml');
$oTemplate->tags
(
    'products' => $itemString
);
$oTemplate->clear;

```

```

print "Content-type: text/html\n\n";
print $oTemplate->get();

```

```

#####
## WriteToFailedDatabase: if the transaction fails write it to the failed
## _transactions table
#####

```

```
sub WriteToFailedDatabase($,\%,$, $)
```

```

{
    my $sTransCode = shift;
    my $hDash = shift;
    my $sUserName = shift;
    my $oDBH = shift;

    my %transInfo;
    ##write transaction info into database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, $oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

    $transInfo{'user_seq'} = $oDiskAccount->fetchColumn('USER_SEQ');
    $oDiskAccount->finish();
    $transInfo{'trans_code'} = $sTransCode;
    $transInfo{'product_code'} = $hDash->{'pi_code'};
    $transInfo{'IP'} = $hDash->{'ipaddr'};

    my $intoDB = XDrive::Sale::Purchase->new($oDBH);
    my $error = $intoDB->FailedTransaction(\%transInfo);

    return $error;
}

```

```

#####
## WriteToPurchaseDatabase: write the user transaction info to th
user_purchase
## table
#####

```

```
sub WriteToPurchaseDatabase($,\%, $, $)
```

```

{
    my $sTransCode = shift;
    my $hDash = shift;
    my $sUserName = shift;
    my $oDBH = shift;

    my %transInfo;
    ##write transaction info into database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

    $transInfo{'user_seq'} = $oDiskAccount->fetchColumn('USER_SEQ');
    $transInfo{'account_seq'} = $oDiskAccount->fetchColumn('USER_SEQ');
    $oDiskAccount->finish();
    $transInfo{'trans_code'} = $sTransCode;
    $transInfo{'product_code'} = $hDash->{'pi_code'};

    my $intoDB = XDrive::Sale::Purchase->new($oDBH);
    my $error = $intoDB->Checkout(\%transInfo);
    return $error;
}

```

```

#####
## TransactionOK: if the tranaction was processed and ok'ed, we add the
proper space to the
## user's xdrive and let them know the space has been added
#####

```

```

sub TransactionOK($,\%,$,$)

```

```

{
    my $sTransCode = shift;
    my $hDash = shift;
    my $sUserName = shift;
    my $oTemplate = shift;
    my $oDBH = shift;
    my $oToken = shift;
    my $oCGI = shift;
    my $oErr = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);
    my $userSeq= $oDiskAccount->fetchColumn('USER_SEQ');

    my @aCodes=split(/\|/, $sTransCode);
    $aCodes[1]=~s/~//;
    my $sNewQuota;
    my $sAddedSpace;

    my $oDeal = XDrive::DatabaseO::Table::Deal->new(undef,$oDBH);
    $oDeal->loadWhere('PRODUCT_CODE', $hDash->{'pi_code'});
    my $itemSeq = $oDeal->fetchColumn('ITEM_SEQ');
    my $oItem = XDrive::DatabaseO::Table::Item->new(undef,$oDeal-
>fetchDBO());
    $oItem->loadWhere('SEQ', $itemSeq);
    my $sCode = "EPOCH_" . $oItem->fetchColumn('CODE');
    my $sDescription = $oErr->ReturnMessageGivenCode($sCode);

    my $sSpaceToAdd = $oItem->fetchColumn('NAME');

    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);
    $sNewQuota = $sSpaceToAdd + $oAction->QuotaLimit();
}

```

```

##now set the new quota
##in the database and in the ncftpd database
##used during testing to reset occasionally
##$sNewQuota = 25600;
XDQuotaLimit($sUserName, $sNewQuota);

##insert into the spool to update ftp account

## Load the required template HTML files.
$soTemplate->load('addspace_ok.html');
$soTemplate->tags
(
  (
    'transactionCode' => $aCodes[1],
    'addedSpace' => $sDescription
  );
$soTemplate->clear;
print "Content-type: text/html\n\n";
print $soTemplate->get();
)

#####
## TransactionBad: If we get an error code beginning with and N, it's a
declined transaction
## get the error code and give user the bad transaction page with error code
#####

sub TransactionBad($,$)
{
  my $sTransCode = shift;
  my $soTemplate = shift;
  my $soErrors = shift;

  if ($sTransCode!~/^\d+$/ )
  {
    ##error codes contains
    $sTransCode="EPOCH_" . $sTransCode;
    chop($sTransCode);
  }

  ##$soErrors->AddErrorByErrorCode($sTransCode);

  $soErrors->AddErrorByCodeIncludes($sTransCode);
  my $sReturnError=$soErrors->Message($sTransCode);

  if(!$sReturnError)
  {
    $sReturnError = "The was an problem processing your transaction.
Please try again.";
  }

  ## Load the required template HTML files.
  $soTemplate->load('addspace_bad.html');
  $soTemplate->tags
  (
    (
      'error' => $sReturnError
    );
  );
  $soTemplate->clear;
  print "Content-type: text/html\n\n";
  print $soTemplate->get();
}

```

###client_info.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});
use CGI;

exit &main;

sub main ()
{
    my $oCGI = CGI->new();

    ##get this info from Michael Ryan's or Gavin's client
    my $sUsername      = $oCGI->param('username');
    my $sClientType     = $oCGI->param('client_type');
    my $sClientVersion = $oCGI->param('client_version');
    my $bFirstTime      = $oCGI->param('first_time');

    ##hash of NT info for current version of client
    ##version 1.0 is 0 in the array of upgrades
    my %infoNT;
    my @featuresNT;
    $infoNT{'current_version'} = '1.0';
    $infoNT{'force_upgrade'} = 0;
    $infoNT{'client_url'} = 'http://www.xdrive.com/download/xdrivent.exe';
    ##holds the first array subscript in which upgrade info is kept
    $infoNT{'1.0'} = 0;
    $featuresNT[0][0] = 'beta release';
    ## $featuresNT[0][1] = 'First new feature';
    ## $featuresNT[0][2] = 'Second new feature';

    ##hash of 95 info for current version of client
    ##version 2.03 is 0 in the array of upgrades
    my %info95;
    my @features95;
    $info95{'current_version'} = '2.03';
    $info95{'force_upgrade'} = 0;
    $info95{'client_url'} = 'http://www.xdrive.com/download/xdrive.exe';
    $info95{'2.00'} = 0;
    $info95{'2.01'} = 1;
    $info95{'2.02'} = 2;
    $info95{'2.03'} = 3;
    $info95{'2.04'} = 4;
    $features95[3][0] = 'automatic proxy support.';

    ## examples of other features
    ## $features95[0][1] = '2.03 feature 1';
    ## $features95[0][2] = '2.03 feature 2';
    ## $features95[1][0] = '2.04 feature 1';
    ## $features95[1][1] = '2.04 feature 2';

    my $returnString='';
    my $ref_to_hash;
    my $ref_to_array;

    ##point to hash and array for type of client
    ##this way no need to create separate functions
    if ($sClientType =~ /^xdwin9x/)
    {
        $ref_to_hash=\%info95;
    }
}
```

```

        $ref_to_array=\@features95;
    }
    elsif ($sClientType =~ /^xdwinnt/)
    {
        $ref_to_hash=\%infoNT;
        $ref_to_array=\@featuresNT;
    }
    else {}

    if (($sClientType =~ /^xdwin9x/) || ($sClientType =~ /^xdwinnt/))
    {
        ##if the user's version of the client is older than the
        ##current version, ask them to upgrade and tell them
        ##about new features
        my $feature_text='';
        if ($ref_to_hash->{'current_version'} > $sClientVersion)
        {
            ##get all features from the version 1 above the user's
            ##to the current version
            my $array_number_start = $ref_to_hash->{$sClientVersion} +
1;
            my $array_number_end = $ref_to_hash->{$ref_to_hash-
>{'current_version'}};
            ##Assemble a big string of new features for
            ##newer versions than user has
            my ($i,$j);
            for $i ($array_number_start .. $array_number_end)
            {
                for $j (0 .. $#{$ref_to_array->[$i]})
                {
                    $feature_text .= " - ".$ref_to_array->[$i][$j]
. "|";
                }
            }

            $returnString = join ("\n",
"client_version=$ref_to_hash->{'current_version'}",
"force_upgrade=$ref_to_hash->{'force_upgrade'}",
"client_url=$ref_to_hash->{'client_url'}",
"client_text=$feature_text",
);
        }
        else
        {
            $returnString = join ("\n",
"client_version=0.0",
"force_upgrade=-1",
"client_url=No url. Please contact X:drive",
"client_text=",
);
        }

        print $oCGI->header();
        print $returnString;

        ##if ($bFirstTime)
        ##
        ##    ## Record the version number
        ##    ## XDClientFirstTimeUse
        ##    ##    (
        ##    ##        $sUsername,
        ##    ##        $sClientType,

```

WO 01/33381

PCT/US00/30536

```
## $sClientVersion
## );
## }
}
```

###cookie.cgi

```
#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> to verify that the user is
# good to login, if they are then log them in and otherwise redirect to
# a not authorized page.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::UserQuota;
use XDrive::DatabaseO::Table::Language;
use XDrive::DatabaseO::Search;

use CGI;
use XDrive::CGI::Cookie;
use CGI::Carp qw(fatalsToBrowser);

use XDrive::CGI;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Template;
use XDrive::Library;
use XDrive::DatabaseO;
use Mail::Sendmail;

&main;
exit;

sub main
{
    my $oCGI      = new CGI;
    my $oErr      = new XDrive::Error;
    my $oDBO      = new XDrive::DatabaseO;
    my $oCookie   = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    my $oToken;
    my $sToken;
    my $sUsername;
    my $sPartnerCode;

    my $bSecurity = $oCGI->param('bSecurity');
    my $sPartnerToken = $oCGI->param('partner_token');

    my $passed_lang = $oCGI->param('language');

    #####
    ## Attempt to authenticate the user by using one of the following two
    ## authentication methods: username/password pair or partner token
    ## authentication.
    #####
    if (! defined $sUsername && length($sPartnerToken) > 20)
    {
        authPartnerUser($oCGI,$oErr,$oDBO,\$sUsername,\$oToken,
            \$sPartnerCode,$sPartnerToken);
        $sToken = $oToken->name();
    }
    else
    {
        authWebSiteUser($oCGI,$oErr,$oDBO,\$sUsername,\$oToken);
    }
}
```



```

        $sPartnerCode = 'xdrv';
    }

#####
## If an error occurred while trying to create a token then redirect
## the user to the error page.
#####
if ($oErr->Occurud)
{
    $oDBO->disconnect;
    xd_fatal_error($oCGI,$oErr);
    exit;
}

#####
## If we have gotten here then we have an authenticated user.
#####

#####
## Build and print out cookies
#####
my $sLanguage = getLanguage($oDBO,$sUsername);

##check if user's language is the same as passed language
if ((length($passed_lang) > 0) && $sLanguage ne $passed_lang)
{
    ##update db here to new language
    setLanguage($oDBO,$sUsername,$passed_lang);
    ##update session to new language
    $sLanguage = $passed_lang;
}

##delete the promo cookie; this will not be set here and we
##don't want an old one hanging out
##promo cookies should be set in promo.cgi
$oCookie->deleteElement('promo') if $oCookie->getElement('promo');

$oCookie->setElement
(
    (
        'language' => $sLanguage,
        'partner' => $sPartnerCode,
    )
);

print "Set-Cookie: ". $oCookie->asString();
print "Set-Cookie: SST=$sToken; domain=xdrive.com; path=/\n"
    if $sPartnerCode ne 'xdrv';

#####
## write user login to the database
#####
&incrementLoginNumber($oDBO,$sUsername,$sLanguage,$sPartnerCode);

#####
## Send the user off into thier file explorer
#####
if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
{
    print $oCGI->redirect("?sst=".$oToken->name()."&sid=0");
}
else

```

```

    {
        xd_web_open($oCGI, "", "", \%ENV, $bSecurity);
    }

    $oDBO->disconnect;
    return 0;
}

sub incrementLoginNumber()
{
    my $oDBO = shift;
    my $sUsername = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $timesLoggedIn = $oDiskAccount->fetchColumn("LOGIN_NUM");
    my $user_seq = $oDiskAccount->fetchColumn("USER_SEQ");

    if ($timesLoggedIn)
    {
        $timesLoggedIn++;
    }
    else
    {
        $timesLoggedIn=1;
    }

    $oDiskAccount->setColumn("LOGIN_NUM", $timesLoggedIn);
    $oDiskAccount->setColumn("LAST_LOGIN",XToday());

    my $status = $oDiskAccount->update();

    if ($status > -1)
    {
        $oDiskAccount->commit();
        $oDiskAccount->finish();

        ##give user extra 10MB if 10th login
        if ($timesLoggedIn == 10)
        {
            my $oUserQuota = XDrive::DatabaseO::Table::UserQuota-
>new(undef,$oDBO);
            $oUserQuota->loadWhere("USER_SEQ", $user_seq);
            my $additional_quota = $oUserQuota-
>incrementQuota($user_seq,10240);
            if ($additional_quota > 0)
            {
                &send_email($user_seq, $oDBO,
$additional_quota,$sLanguage, $sPartnerCode);
            }
        }
    }
    else
    {
        # $oDiskAccount->rollback();
    }
}
}

```

```

sub send_email
{
    my $user_seq = shift;
    my $oDBO = shift;
    my $additional_quota = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    ##comes in as k, change to megabytes
    my $mbs = $additional_quota/1024;

    my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef,$oDBO);
    $oUserData->loadWhere("SEQ", $user_seq);
    my $email_address = $oUserData->fetchColumn("EMAIL_ADDRESS");
    my $name_first = $oUserData->fetchColumn("NAME_FIRST");
    my $name_last = $oUserData->fetchColumn("NAME_LAST");

    my $oTemplate = new XDrive::Template
    (
        (
            'language' => $sLanguage,
            'partner_code' => $sPartnerCode,
        )
    );

    $oTemplate->load('received_10MB_10logins.shtml');
    $oTemplate->tags(
        (
            'mbs' => $mbs,
        )
    );
    $oTemplate->clear();
    my $message = $oTemplate->get;

    my %toXdrive =
    (
        To => "$name_first $name_last <$email_address>",
        Bcc => '',
        From => "support@xdrive.com",
        Message => $message,
        Subject => "Congratulations!"
    );

    sendmail(%toXdrive);
}

sub authPartnerUser
{
    my $oCGI = shift;
    my $oErr = shift;
    my $oDBO = shift;
    my $rsUsername = shift;
    my $roToken = shift;
    my $rsPartnerCode = shift;
    my $sPartnerToken = shift;

    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    my $oPartnerToken = new Token
    (
        (
            'err' => $oErr,
            'dbh' => $oDBO,
        )
    );
    $oPartnerToken->load($sPartnerToken);

    return if $oErr->Occurud;
}

```

```

    $$roToken = new Token
    (
        'dbh' => $oDBO,
        'err' => $oErr,
        'user_sequence' => $oPartnerToken->data('user_seq'),
    );
    $$roToken->create();

    return if $oErr->Occurud;

    ### Edited by Justin so that the partner_code is looked for
    ### in the cookie instead of the token table.
    $$rsPartnerCode = $oPartnerToken->data('partner_code');
    $$$rsPartnerCode = $oCookie->getElement('partner');
    $$rsUsername = $oPartnerToken->data('user');

    $$roToken->data('ip', $ENV{REMOTE_ADDR});
    $$roToken->data('browser', $ENV{HTTP_USER_AGENT});
    $$roToken->data('user', $$rsUsername);
    $$roToken->data('user_seq', $oPartnerToken->data('user_seq'));
    $$roToken->data('partner_code', $$rsPartnerCode);
    $$roToken->data('disk_account_seq', $oPartnerToken-
>data('disk_account_seq'));
    $$roToken->save;

    $oPartnerToken->delete();
}

sub authWebSiteUser
{
    my $oCGI = shift;
    my $oErr = shift;
    my $oDBO = shift;
    my $rsUsername = shift;
    my $roToken = shift;

    my $sPassword = $oCGI->param('pass');
    $$rsUsername = $oCGI->param('user');

    $oCGI->param('user');

    if (xd_auth_password($$rsUsername, $sPassword, $oDBO))
    {
        ## Login the user info X:drive and get the session token
        $$roToken = xd_login($oCGI, $$rsUsername, $oErr, $oDBO);
    }
    else
    {
        $oErr->AddErrorByErrorCode('501');
    }
}

sub getLanguage
{
    my $oDBO = shift;
    my $sUsername = shift;

    my $language;

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, $oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);

```

```

    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings-
>new(undef,$oDBO);
    $oUserSettings->loadWhere("USER_SEQ",$userSeq);
    $oUserSettings->finish;
    my $language = $oUserSettings->fetchColumn("LANGUAGE");

    if ($language eq '')
    {
        $language = 'english';
    }
    else
    {
        ## Get language from database given code
        my $oLanguage = XDrive::DatabaseO::Table::Language-
>new(undef,$oDBO);
        $oLanguage->loadWhere("SEQ",$language);
        $oLanguage->finish;
        $language = $oLanguage->fetchColumn("CODE");
    }

    return $language;
}

sub setLanguage
{
    ##set the LANGUAGE column of the User_Settings table to passed
    language

    my $oDBO = shift;
    my $sUsername = shift;
    my $language = shift;

    my ($rv,$errorCode);

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBO);
    ##grab right table
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings-
>new(undef,$oDBO);
    $oUserSettings->loadWhere("USER_SEQ",$userSeq);
    $oUserSettings->finish;

    ##grab the seq number of the LANGUAGE being passed
    my $oLanguage = XDrive::DatabaseO::Table::Language->new(undef,$oDBO);
    $oLanguage->loadWhere("CODE",$language);
    $oLanguage->finish();
    my $seq_lang = $oLanguage->fetchColumn("SEQ");

    eval
    {
        ##
        ##set language here
        $rv = 0;
    }
}

```

```
$oUserSettings->setColumn('LANGUAGE',$seq_lang);
$rv = $oUserSettings->update();
};
if ($rv == 0)
{
    $oUserSettings->rollback();
    $errorCode = 0;
}
else
{
    $oUserSettings->commit();
    $errorCode = 1;
}
return $errorCode;
}
```

###download_client.cgi

```

#!/usr/bin/perl
## Written by Karen Eppinger
## Script that shows the 'download the client' page
## it can no longer be static html because we need to
## do some checking on whether the user is from a partner or not
## if so, make sure to let them know what their X:drive login name
## is if it differs from their partner login

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use XDrive::Library;
use XDrive::Template;
use XDrive::Error;
use XDrive::DatabaseO;
use XDrive::Client::Security;
use XDrive::DatabaseO::Table::ResellerUserMap;
use XDrive::DatabaseO::Table::Reseller;

&main;
exit;

sub main
{
    ## Load the session token
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;
    my $oCGI = new CGI;

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        $oDBO->disconnect();
        exit;
    }

    my $partner_code = $oToken->data('partner_code');
    my $language = $oCookie->getElement('language') || 'english';

    my $oForm = new XDrive::Template
    (
        (
            'partner_code' => $partner_code,
            'language' => $language,
        )
    );

    $oForm->load('download_client.shtml');

    ##if we are coming from a partner, make sure partner login
    ##and X:drive login match

    my $reseller_username;
    my $reseller_name;
    my $partner_warning;
    my $username;

```

```

    if ($partner_code ne 'xdrv')
    {
        my $user_seq = $oToken->data('user_seq');
        $username = $oToken->data('user');

        my $oResellerUserMap = XDrive::DatabaseO::Table::ResellerUserMap-
>new(undef, $oDBO);
        my $oReseller = XDrive::DatabaseO::Table::Reseller->new(undef,
$oDBO);
        $oReseller->loadWhere("CODE", $partner_code);
        $reseller_name = $oReseller->fetchColumn("NAME");
        $oResellerUserMap->loadWhere("USER_SEQ", $user_seq);
        $reseller_username = $oResellerUserMap->fetchColumn("ALIAS");

        if ($reseller_username ne $username)
        {
            ##load the text for the warning message
            my $oWarning = new XDrive::Template
            (
                {
                    'partner_code' => $partner_code,
                    'language' => $language,
                }
            );

            $oWarning->load('download_client_warning.shtml');
            $oWarning->tags
            (
                {
                    'reseller_name' => $reseller_name,
                    'reseller_username' => $reseller_username,
                    'username' => $username,
                }
            );

            $oWarning->clear();
            $partner_warning = $oWarning->get();
        }
    }

    $oForm->tags
    (
        {
            'partner_warning' => $partner_warning,
            'reseller_name' => $reseller_name,
            'reseller_username' => $reseller_username,
            'username' => $username,
        }
    );

    $oForm->clear();

    print $oCGI->header(), $oForm->get;

    $oDBO->disconnect();

    return 0;
}

```


###email_change.cgi

```

#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::DiskAccount;

use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Library;
use XDrive::Template;
use XDrive::Security;
use XDrive::CGI;
use XDrive::Error;

use strict;

&main;
exit;

sub main
{
    my $oCGI = CGI->new();
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    #####
    ## Check the token is valid and is an error occurred then
    ## redirect with a fatal error
    #####

    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    my $sUserName = $oToken->data('user');
    my $sOldEmail = $oCGI->param('oldEmail');
    my $sNewEmail = $oCGI->param('newEmail');

    if (($sOldEmail eq '') || ($sNewEmail eq ''))
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1350);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    ##first, get user_seq from the disk_account table
    ##since we only have the user name, need to do this first
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,undef);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);
    my $nUserID = $oDiskAccount->fetchColumn('USER_SEQ');

    ##now that we have that, get the email address from

```

```

##user table using the user_seq number to pull the seq number
my $oUserInfo = XDrive::DatabaseO::Table::UserData->new(undef,
$oDiskAccount->fetchDBO());
$oUserInfo->loadWhere('EMAIL_ADDRESS', $sNewEmail);

##if a sequence number is returned, there is already a record
##in the database with that email address. don't allow to change

my $nSeqNumber = $oUserInfo->fetchColumn('SEQ');

if ($nSeqNumber)
{
    $oUserInfo->disconnect();
    my $sMessage = $oErr->ReturnMessageGivenCode(1351);
    XDErrorToBrowser("", $sMessage, undef, $oToken);
}
else
{
    $oUserInfo->loadWhere('SEQ', $nUserID);
    my $sEmailinDB = $oUserInfo->fetchColumn('EMAIL_ADDRESS');

    if ($sOldEmail eq $sEmailinDB)
    {
        ##set email in class
        $oUserInfo->setColumn('EMAIL_ADDRESS', $sNewEmail);
        ##now update database
        $oUserInfo->update();

        my $oTemplate = new XDrive::Template
            ({'partner_code' => $oToken-
>data('partner_code')});
        $oTemplate->load('pr_changeemail_ok.html');
        print "Content-type: text/html\n\n";
        print $oTemplate->get();
    }
    else
    {
        $oUserInfo->disconnect();
        my $sMessage = $oErr->ReturnMessageGivenCode(1352);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }
}
$oUserInfo->commit();
$oUserInfo->finish();
$oUserInfo->disconnect();
}

```

###error.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});
use XDrive::Error;
use XDrive::Template;
use CGI;

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my ($sErrorCode) = $ENV{QUERY_STRING} =~ /error=([^&\|=]+)/;
    my $oError = new XDrive::Error;
    my $sError = $oError->ReturnMessageGivenCode($sErrorCode);
    my $oTemplate = new XDrive::Template( {'partner_code' => 'xdrv'} );
    $oTemplate->load('generic_error.thtml');
    $oTemplate->tags
        (
            {
                'message' => $sError
            }
        );
    $oTemplate->clear();
    print $oCGI->header(), $oTemplate->get;
}
```

###explorer.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Tue May 25 15:23:31 PDT 1999.
## Program to build the file explorer which is itself a popup window.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
#use vars qw(@ISA);
```

```
#@ISA = qw(XDrive::CGI);
```

```
use CGI qw(param header);
use CGI::Carp qw(fatalsToBrowser);
use Date::Format;
use HTTP::Icons;
# use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::Client::Quota;
use XDrive::Library;
use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::Error;
```

```
&main;
```

```
exit(0);
```

```
sub main
```

```
{
    ####
    ## Global variables
    ####
    my $oToken;          ## XDrive Token
    my $sUsername;       ## username
    my $sPath;           ## path for index
    my $sSST;            ## Token name
    my $bEditExt;        ## Allow extensions to be edited?
    my $bFirstTime;      ## First time the've logged in...
    my $bExtraHelp;      ## Print extra help
    my $bMarketing;      ## does user want to receive offers from other
companies
    my $bNewsletter;     ## does user want to receive our newsletter
    my $sPartner;        ## partners name
    my $g_sFrameSize;    ## breakdown of the centerview frame
    my $g_sFrameBanner;  ## banner view frame information

    my $oDBO = XDrive::DatabaseO->new(undef,undef);
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);
```

```
####
## If the user has bookmarked the X:drive service then redirect
## them back to the homepage
####
```

```

if (! length($oCGI->param('sst')) && ! length($oCGI->cookie('SST')))
{
    print $oCGI->redirect('/cgi-bin/web_unauthorized.cgi?error=804');
    $oDBO->disconnect();
    return 0;
}

####
## Check the security and if an error occurs
####
$oToken = xd_security_check($oDBO,$oCGI,$oErr);

if ($oErr->Occurud) {
    $oDBO->disconnect();
    xd_fatal_error($oCGI, $oErr);
    exit;
}

####
## Now we know we have a valid session so pull the partner name
## from a cookie if available or clear the variable
####
# $sPartner = $oToken->data('partner_code');
# $sPartner = $oCookie->getElement('partner');
# $sPartner = "xdrv" if ($sPartner eq "");

## Load the required template HTML files.
# my $oFrame = new XDrive::Template
#     ({
#         'partner_code' => $oToken->data('partner_code')
#     });

### Edited by Justin to check the cookie instead of
### the token table for the partner_code.
my $oFrame = new XDrive::Template
    ({
        'partner_code' => $oCookie->getElement('partner')
    });

## If the request comes from the windows app the give back a simplified
template
$oFrame->load("acct_explorer_frame.shtml");

## Assign globally used variables
$sPath = $oCGI->param('sFolderCurrent');
$sSST = $oToken->name;
$sUsername = $oToken->data('user');

## User settings
my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new(undef,
$oDBO);

my $oUserSettings = XDrive::DatabaseO::Table::UserSettings->new(undef,
$oDBO);
my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef, $oDBO);

$oDiskAccount->loadWhere("USERNAME", $sUsername);
$oUserSettings->loadWhere("USER_SEQ", $oDiskAccount-
>fetchColumn("USER_SEQ"));

```

```

$UserData->loadWhere("SEQ", $oDiskAccount->fetchColumn("USER_SEQ"));

$bEditExt      = $oUserSettings->fetchColumn("FILE_EXT_EDITABLE") == 1 ?
'true' : 'false';
$bExtraHelp    = ($oUserSettings->fetchColumn("EXTRA_HELP") == 1) ?
'true' : 'false';
$bMarketing    = $oUserSettings->fetchColumn("OPT_MARKETING") == 1 ?
'true' : 'false';
$bNewsletter   = $oUserSettings->fetchColumn("OPT_NEWSLETTER") == 1 ?
'true' : 'false';

my $firstName = $UserData->fetchColumn("NAME_FIRST");
my $lastName  = $UserData->fetchColumn("NAME_LAST");

my $first = $oCGI->param('first');
$bFirstTime = $first eq 'yes' ? 'true' : 'false';

## Frame settings
if ($sPartner eq 'cc' || $sPartner eq 'qupa')
{
    $g_sFrameSize   = '100%';
    $g_sFrameBanner = '';
}
else
{
    $g_sFrameSize   = '103,*';
    $g_sFrameBanner = '<FRAME NAME="banner" ' .
        ' SRC="/cgi-bin/ads.cgi" SCROLLING=NO BORDER=0 ' .
        ' FRAMEBORDER=0 MARGINWIDTH=0 MARGINHEIGHT=0 ' .
        ' TOPMARGIN=0 LEFTMARGIN=0>';
}

##get the language information from the cookie
##if no cookie or not set, set to english
my $session_info = $oCGI->cookie('x_session_info');
my $language;

if ($session_info{'language'} ne '') {
    $language = $session_info{'language'};
}
else {
    $language = 'english';
}

my $clientDownload = $oCGI->param('client');
my $sCenterPage = 'centerview.thtml';
if ($clientDownload eq 'getclient') {
    $sCenterPage = 'download_client.thtml';
}

## Set the token name and session ID in the navigation form so that
popup
## windows have access to them and the do not need to be passed around.
$oFrame->tags
(
    'sSST' => $sSST,
    'bSettingEditExtensions' => $bEditExt,
    'sPartner' => $sPartner,
    'bExtraHelp' => $bExtraHelp,
    'bFirstTime' => $bFirstTime,
    'bMarketing' => $bMarketing,
    'bNewsletter' => $bNewsletter,
    'centerPage' => $sCenterPage,

```

```
'userName' => $sUsername,  
'firstName' => $firstName,  
'lastName' => $lastName,  
'frameBanner' => $g_sFrameBanner,  
'frameSize' => $g_sFrameSize,  
'language' => $language  
});  
  
## Print out the HTML and exit  
$oFrame->clear();  
print $oCGI->header(), $oFrame->get;  
  
$oDiskAccount->finish();  
$oUserSettings->finish();  
$oUserData->finish();  
  
$oDBO->disconnect();  
}
```

###explorer_user_data.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Tue May 25 15:23:31 PDT 1999.
## Program to build the file explorer which is itself a popup window.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use vars qw(@ISA);

@ISA = qw(XDrive::CGI);

use Data::Dumper;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use Token;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::Library;
use XDrive::Template;
use XDrive::Error;

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    my $sFolder;
    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);

    $sFolder = $oCGI->param('folder_current');

    ## Load the required template HTML files.
    my $oFrame = new XDrive::Template
    (
        ('partner_code' => $oToken->data('partner_code'))
    );

    if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
    {
        $oFrame->load("acct_user_data_xd_win.thtml");
    }
    else
    {
        $oFrame->load("acct_user_data.thtml");
    }
}
```



```
## Set the token name and session ID in the navigation form so that
popup
## windows have access to them and they do not need to be passed around.
$oframe->tags
  (
    'sst' => $oaction->SST(),
    'sid' => $oaction->SID(),
    'usage_total' => $oaction->QuotaLimit(),
    'usage_used' => $oaction->QuotaUsed(),
    'stuff' => $oaction->DiskAccountXML($sFolder)
  );
$oframe->clear;

$oaction->DisconnectDB();

## Print out the HTML and exit
print "Cache-Control: no-cache\n";
print "pragma: no-cache\n";
print "Content-type: text/html\n\n";
print $oframe->get;
}
```

###file_load.cgi

```

#!/usr/bin/perl
# Program written by Martin Hald <mhald@uci.edu> to fetch files from a
# storage area or database and return them via a HTTP socket to the user.

use strict;
use CGI qw(header param);
use CGI::Carp 'fatalsToBrowser';

## The HTTP::MimeTypes module was a quick module that I wrote that reads the
## standard apache mime.types file, parses it and given any known extension
## translates it to the correct mimetype.

use lib ($ENV{PERL_XDRIVE_LIB});

use HTTP::MimeTypes;
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::DatabaseO::Table::DiskItemShare;
use XDrive::DatabaseO;
use XDrive::Library;
use XDrive::Error;

## We have two security methods when downloading files:
## 1) tokens
## 2) claim checks
## to deal with this we simply security method we are using and process the
## request.

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    my $sFileCurrent; ## Current File
    my $oAction;      ## Action object

    ####
    ## Process the request as a share a file pickup if the claim_check
    ## param is available
    ####
    if (param('claim_check'))
    {
        my $oShare;
        $oShare = XDrive::DatabaseO::Table::DiskItemShare->new();
        $oShare->loadWhere("random_key", param('claim_check'));

        $oAction = new XDrive::Client::Actions($oShare,$oCGI);
        $sFileCurrent = join
            (
                '/',
                $oShare->fetchColumn("ITEM_PATH"),
                $oShare->fetchColumn("ITEM_NAME")
            );
    }
    ####
}

```

```

## Otherwise it is an request from the browser or PC client
## side program so let the actions object handle the request
####
else
{
####
## Attempt to authenticate the user and if that fails
## then redirect to the error CGI
####
my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

if ($oErr->Occurud)
{
    xd_fatal_error($oCGI,$oErr);
    exit;
}

####
## Now we know that we have a valid token so go ahead
## and let the actions object handle the request
####

$oAction = new XDrive::Client::Actions
(
    $oToken,
    $oCGI
);

$oFileCurrent = $oAction->FileCurrent();
}

## Check that the current file is OK. If this check fails then
## the code does an XDErrorToBrowser and exists
$oAction->FileCheck($oFileCurrent);

print _header($oFileCurrent);

## Commented out by Justin because it was
## including a 1 at the end of the file by printing it out.
#print $oAction->FileLoad($oFileCurrent);
$oAction->FileLoad($oFileCurrent);

$oDBO->disconnect;
}

sub _header
{
    my $oFile = shift;
    my $oMlt = new HTTP::MimeTypes;

    ## Grab the extension and lookup the correct mimetype using the mlt or
mime    ## lookup table object.

    my $oHeader;    ## MIME header
    my $oExtension; ## file extension

    ## Clean up the filename by getting rid of any path that comes before
    ## the filename.
    $oFile =~ s=.*/=g;

    if (param('mime') eq 'download')

```

```

        {
            if ($ENV{HTTP_USER_AGENT} =~ /MSIE/)
            {
                $sHeader .= "Content-Disposition: attachment;
filename=$sFile\n";
                $sHeader .= "Content-type: application/download;
name=\"\$sFile\"\\n\\n";
            }
            else
            {
                $sHeader .= "Content-type: application/octet-stream\\n\\n";
            }
        }
    else
    {
        my $dotPos=-1;
        my $returnPos=-1;
        while (($dotPos = index($sFile, ".", $dotPos)) > -1)
        {
            $returnPos = $dotPos;
            $dotPos++;
        }

        ##if no extension set extension to nothing
        if ($returnPos < 0)
        {
            $sExtension='';
        }
        else
        {
            $sExtension = substr($sFile,$returnPos+1);
        }

        $mlt->extension($sExtension);
        $sHeader = $mlt->header();
    }

    return $sHeader;
}

sub IEHack ()
{
    my $sFileCurrent = param('sFileCurrent');
    my ($sFileOnly) = $sFileCurrent =~ /\\[([^\[]+)\]/;
    my $sJavascript;

    if (param('source') eq 'www.fileExplorer.view' || param('source') eq
'www.fileExplorer.download')
    {
        $sJavascript = <<EOM;
<SCRIPT LANGUAGE=JAVASCRIPT>
if (parent.parent.parent.name)
{
    parent.parent.parent.parent.XDReset();
    parent.parent.parent.parent.XDRefreshExplorer();
}
</SCRIPT>
EOM
    }

    print <<EOM;
Content-type: text/html

```

```
<HTML>
<BODY>
```

```
$sJavascript
```

```
<OBJECT classid=CLSID:4CCF6192-4552-11D3-80A8-0050048D4BF8
        codebase="http://209.101.43.96/dll/xdfiles.cab"
        id=XDFiles>
</OBJECT>
```

```
<SCRIPT LANGUAGE="VBSCRIPT">
```

```
' Don't raise errors
On Error Resume Next
```

```
Dim oXDFiles          ' The ActiveX control
```

```
' Late bind to the control
Set oXDFiles = CreateObject("XDFiles.XDFiles.1")
```

```
' If we got an error, they didn't install the ActiveX control
If Err.Number <> 0 Then
    MsgBox "You must install the X:drive ActiveX control in order to
download " & _
    "the file. Please click Download again and when prompted to
install the " & _
    "ActiveX control, click Yes."
End If
```

```
' Set some test values for the properties
oXDFiles.Prompt = True
oXDFiles.Destination = "c:\\$sFileOnly"
oXDFiles.File = "$sFileCurrent"
```

```
' Call each method
oXDFiles.Get
```

```
' Print out each property
' document.write("oXDFiles.Destination = " & oXDFiles.Destination & "<br>")
' document.write("oXDFiles.Prompt = " & CBool(oXDFiles.Prompt) & "<br>")
' document.write("oXDFiles.File = " & oXDFiles.File & "<br>")
' document.write("oXDFiles.ServerSideToken = " & oXDFiles.ServerSideToken &
"<br>")
' document.write("oXDFiles.SessionId = " & oXDFiles.SessionId & "<br>")
```

```
' Free the ActiveX control
Set oXDFiles = Nothing
```

```
</SCRIPT>
```

```
</BODY>
```

```
</HTML>
```

```
EOM
```

```
}
```

###file_save.cgi

```
#!/usr/bin/perl

#####
### file_save.cgi
#####

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
use CGI::Carp qw(fatalsToBrowser);
use Token;

use XDrive::CGI2;                ## file upload functions
use XDrive::CGI qw(:MAIN);      ## xd_web_buttonindex function
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Library;            ## xd_fatal_error function
use XDrive::DatabaseO;
use XDrive::DatabaseO::Search;
use XDrive::Template;
use XDrive::DatabaseO::Transaction;

&main;

exit;

sub main {
    my $oErr    = new XDrive::Error;
    my $oDBO    = new XDrive::DatabaseO;
    my $oSearch = new XDrive::DatabaseO::Search;

    my $oTransaction = XDrive::DatabaseO::Transaction->new($oDBO);

    #####
    ## Parse the SST cookie manually and retrieve the user sequence
    ## by passing it to the getUserSeq sub.
    #####
    my ($cookie) = $ENV{'HTTP_COOKIE'} =~ /\bSST=(\w+)\b/;
    my $user_seq = &getUserSeq($oSearch, $cookie);

    my $bytes = $ENV{'CONTENT_LENGTH'}; ## number of bytes being uploaded.

    my %upload_hash = ('USER_SEQ' => $user_seq,
                       'BYTES'    => $bytes);
    my $oCGI = new XDrive::CGI2(\%upload_hash, $oTransaction);

    #####
    ## Attempt to authenticate the user and if the authentication
    ## fails then redirect to the error CGI
    #####
    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    if ($oErr->Occurud){
        xd_fatal_error($oCGI, $oErr);

        exit;
    }
}
```

```

#####
####
#### Check to see if they've exceeded
#### their quota limit, and error if so.
#####
# my $oUserQuota = XDrive::DatabaseO::Table::UserQuota->new(undef,
$oDBO);
# $oUserQuota->loadWhere("USER_SEQ", $user_seq);
# my $nQuota = $oUserQuota->fetchColumn("QUOTA");
# my $nDiskUsed = $oUserQuota->fetchColumn("DISK_USED");
# if ( ($nQuota * 1024) < ($nDiskUsed + $bytes) ) {
#     $oUserQuota->finish();
#     $oDBO->disconnect();
#     ## let user know he or she has exceeded his quota
#     $oErr->AddErrorByErrorCode(1240);
#     XDErrorToBrowser('action_upload__error.shtml', $oErr, 1,
$oToken);
#     exit(0);
# }
#####
####
## Authentication succeeded so we have a valid session, let
## the actions object handle the request
####
my $oAction = new XDrive::Client::Actions($oToken, $oCGI);

$oAction->SaveUploadedFiles();

####
## File has been uploaded at this point, so set
## the upload inactive in the database.
####
$oTransaction->setUploadInactive();

xd_web_buttonindex($oCGI);
$oAction->DisconnectDB();

$oSearch->disconnect();

return 0;
}

#####
### Subroutine:  getUserSeq
### Parameters:  one object, one scalar
### Returns:    one scalar
### Description: Receives a database search object and an SST token.
###              Queries the token table for the user sequence and returns
it.
#####
sub getUserSeq ($$) {
    my $oSearch = shift;
    my $sst_code = shift;

    my $st = "SELECT user_seq FROM token WHERE code = '$sst_code'";

    my $data = $oSearch->XDSQLSearch($st);

    return $$data[0][0];
}

```

#!/ file_upload_stat.cgi

```
#!/usr/bin/perl

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use XDrive::CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::DatabaseO;
use XDrive::DatabaseO::Search;
use XDrive::Error;
use XDrive::Client::Security;
use XDrive::Template;
use XDrive::Library;
use Token;

&main();

exit(0);

sub main {
    my $oCGI = new CGI;
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    #####
    ### Security Check
    #####
    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);
    if ($oErr->Occurred || (! $oToken)) {
        XErrorToBrowser("", "Security Violation: No token", undef, $oToken);
    }

    my ($tmp_file, @stat_array, $stat_bytes, $meta_refresh, $percent,
$width_green, $width_red);
    my ($url, $tmp_file_string, @tmp_file_array, $error_code);

    my $tmp_path = XDFileUploadTempDir;

    my $oTemplate = new XDrive::Template( {'partner_code' => 'xdrv',
                                           'file'           =>
'file_upload_status.shtml' });

    my $id                = $oCGI->param('id');                ## unique
upload id
    my $nof                = $oCGI->param('nof');                ## number of
files
    my $nof_queried        = $oCGI->param('nof_queried');        ## nof
retrieved from db
    my $file_param         = $oCGI->param('tmp_file');          ## initial file
string
    my $total_file_string = $oCGI->param('total_file_string'); ## string of
all files
    my $param_uploaded     = $oCGI->param('uploaded');          ## bytes
uploaded
    my $bytes              = $oCGI->param('bytes');              ## total number
of bytes

    print $oCGI->header();

```



```
#####
### First, if we're passed an upload id and no temp file params (files to
stat),
### then we either haven't queried the database yet and need to or need to
### query the database again because the number of files (nof) being
uploaded
### is greater than the number of files that our first database query
returned.
#####
```

```
if ($id && (! $file_param)) {
.   ### If this is the first pass, then percent will be a space and width
will be 0.
    $percent = $bytes ? int(100 * ($param_uploaded / $bytes)) : '&nbsp;';
    $width_green = ($percent eq '&nbsp;') ? 0 : $percent;
    $percent .= '%' unless $percent eq '&nbsp;';

    my $seconds;

    $width_red = &width_red($width_green);

    $oTemplate->tags( ('width_green' => $width_green,
                     'width_red'   => $width_red,
                     'percent'     => $percent));

    my $oSearch = new XDrive::Database0::Search($oDBO);

    my ($cnt, $data) = $oSearch->uploadStatusSearch($id);

    ### If no rows were returned from the database, then redirect
    ### and re-query the database.
    if ($cnt == 0) {
        $oSearch->disconnect();

        $seconds = 0;

        $url = "/cgi-bin/file_upload_stat.cgi?" .
            "id=$id&nof=$nof&bytes=$bytes&uploaded=$param_uploaded";

        $meta_refresh = &buildMetaRefresh($seconds, $url);

        &connectingToServer($meta_refresh, $oTemplate);

        exit(0);
    }
    else {
        my $i = 0;

        $bytes      = $$data[$i][0];
        $error_code = $$data[$i][2];

        foreach (@$data) {
            $tmp_file = $$data[$i][1];

            push @tmp_file_array, $tmp_file;

            $i++;
        }

        $tmp_file_string = join '~', @tmp_file_array;

        if ($cnt == $nof) {
            $oSearch->disconnect();
        }
    }
}
```

```

        &statFilesTotal($bytes, $tmp_file_string, $oTemplate);

        exit(0);
    }

    $seconds = 0;

    $url = "/cgi-bin/file_upload_stat.cgi?" .
        "id=$id&nof=$nof&uploaded=$param_uploaded&" .
        "nof_queried=$cnt&bytes=$bytes&tmp_file=$tmp_file_string";

    $meta_refresh = &buildMetaRefresh($seconds, $url);

    my $bytes_uploaded = ($param_uploaded > 0) ? $param_uploaded : '-';

    &redirect($meta_refresh, $bytes_uploaded, $bytes, $oTemplate);

    $oSearch->disconnect();

    exit(0);
}

}
elseif ($file_param) {
    $oDBO->disconnect();

    my @file_array = split '~', $file_param;
    my $ary_cnt = @file_array;

    my ($uploaded_bytes, $seconds);

    if (scalar @file_array > 0) {
        foreach (@file_array) {
            @stat_array = stat("$tmp_path/$_");
            $stat_bytes = $stat_array[7];

            $uploaded_bytes += $stat_bytes;
            push @tmp_file_array, $_;
        }
        if ( ($uploaded_bytes == $param_uploaded) && ($nof > $nof_queried) )
        (
            $seconds = 0;

            $url = "/cgi-bin/file_upload_stat.cgi?" .
                "id=$id&nof=$nof&bytes=$bytes&uploaded=$param_uploaded";

            $meta_refresh = &buildMetaRefresh($seconds, $url);

            $percent = ($bytes == 0) ? 0 : int(100 * ($param_uploaded /
$bytes));
            $width_green = $percent;
            $percent .= '%';

            &redirect($meta_refresh, $uploaded_bytes,
                $bytes, $oTemplate, $percent, $width_green);

            exit(0);
        }
    }
    else {
        $tmp_file_string = join '~', @tmp_file_array;
    }
}
}

```

```

$percent = ($bytes == 0) ? 0 : int(100 * ($uploaded_bytes / $bytes));
$width_green = $percent;
$percent .= '%';

$percent = '&nbsp;' if $width_green == 0;

$seconds = 2;

$url = "/cgi-bin/file_upload_stat.cgi?" .
        "id=$id&bytes=$bytes&nof=$nof&nof_queried=$nof_queried&" .
        "uploaded=$uploaded_bytes&tmp_file=$tmp_file_string";

$meta_refresh = &buildMetaRefresh($seconds, $url);

&redirect($meta_refresh, $uploaded_bytes, $bytes, $oTemplate, $percent,
$width_green);

    exit(0);
}
elseif ($total_file_string) {
    $oDBO->disconnect();
    &statFilesTotal($bytes, $total_file_string, $oTemplate);
}
else {
    $oDBO->disconnect();

    &closeWindow($oTemplate);

    exit(0);
}
}

sub statFilesTotal ($$$) {
    my ($bytes, $tmp_file_string, $oTemplate) = @_;

    my $tmp_path = XDFFileUploadTempDir;

    my @file_array = split '~', $tmp_file_string;

    my (@tmp_file_array, $uploaded_bytes, @stat_array, $stat_bytes);

    my $file_cnt = 0;

    foreach (@file_array) {
        if (-e "$tmp_path/$_") {
            @stat_array = stat("$tmp_path/$_");
            $stat_bytes = $stat_array[7];

            $uploaded_bytes += $stat_bytes;

            push @tmp_file_array, $_;

            $file_cnt++;
        }
    }

    if ($file_cnt == 0) {
        &closeWindow($oTemplate);

        exit(0);
    }
    else {
        my $percent = int(100 * ($uploaded_bytes / $bytes));

```

```

    my $width_green = $percent;

    $percent .= '%';

    $percent = '&nbsp;' if $width_green == 0;

    my $seconds = 2;

    my $url = "/cgi-bin/file_upload_stat.cgi?" .
        "bytes=$bytes&total_file_string=$tmp_file_string";

    my $meta_refresh = &buildMetaRefresh($seconds, $url);

    &redirect($meta_refresh, $uploaded_bytes, $bytes, $oTemplate, $percent,
$width_green);

    exit(0);
}
}

sub redirect ($$$$;$) {
    my ($meta_refresh, $bytes_uploaded, $bytes, $oTemplate, $percent,
$width_green) = @_;

    if ($bytes > 1024) {
        $bytes = sprintf "%.f", ($bytes / 1024);
        $bytes .= 'k';
    }

    if ($bytes_uploaded > 1024) {
        $bytes_uploaded = sprintf "%.f", ($bytes_uploaded / 1024);
        $bytes_uploaded .= 'k';
    }

    my $width_red = &width_red($width_green);

    $oTemplate->tags( {'meta_refresh' => $meta_refresh,
        'bytes_uploaded' => $bytes_uploaded,
        'bytes_total' => $bytes,
        'percent' => $percent,
        'width_green' => $width_green,
        'width_red' => $width_red} );

    $oTemplate->clear();

    print $oTemplate->get;
}

sub closeWindow ($) {
    my $oTemplate = $_[0];

    $oTemplate->load('file_upload_stat__window_close.shtml');

    print $oTemplate->get;
}

sub connectingToServer ($$) {
    my ($meta_refresh, $oTemplate) = @_;

    $oTemplate->load('file_upload__connecting.shtml');

    $oTemplate->tags( {'meta_refresh' => $meta_refresh} );
}

```

```
    print $oTemplate->get;
}

sub buildMetaRefresh ($$) {
    my ($seconds, $url) = @_;

    my $meta_refresh = "<meta http-equiv=refresh content=\"\$seconds;
url=$url\">";

    return $meta_refresh;
}

sub width_red {
    my $width_green = shift;
    my $width_red = ((100 - $width_green) > 0)? 100 - $width_green : 0;

    return $width_red;
}
```

###folder_create.cgi

```

#!/usr/bin/perl
# Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999.

use strict;
use vars qw(@ISA);
use lib ($ENV{PERL_XDRIVE_LIB});
#use lib qw(/export/home/xdrive/lib);

$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)};    # Make %ENV safer

@ISA = qw(XDrive::CGI);

use CGI::Carp 'fatalsToBrowser';
use Date::Format;
use Token;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::Client::Actions;

use CGI;
use XDrive::DatabaseO;
use XDrive::Error;

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    ####
    ## Attempt to authenticate the user
    ####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    ####
    ## If the authentication failed then redirect to the
    ## error cgi and exit
    ####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    ####
    ## Otherwise we know that we have a valid session and
    ## can continue normally
    ####

    my $oAction = new XDrive::Client::Actions
    (
        $oToken,
        $oCGI
    )

```

);

```
$oAction->FolderCreate();  
xd_web_buttonindex($oCGI);  
$oAction->DisconnectDB();  
return 0;  
}
```

###forgot_password.cgi

```
#!/usr/bin/perl
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
```

```
use CGI qw(param header);
use CGI::Carp qw(fatalsToBrowser);
use Token;
use XDrive::CGI ();
use XDrive::Template;
use XDrive::Client::Registration;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Search;
use XDrive::Library;
use XDrive::Utils::RandomString;
```

```
use Mail::Sendmail;
```

```
use constant TRUE => (1==1);
use constant FALSE => ! TRUE;
```

```
#####
```

```
my $request_template = "forgot_password_request.html";
my $thank_you_template = "forgot_password_t_y.html";
my $alert_template = "forgot_password_alert.html";
my $email_template = "password_admin_email.html";
#####
```

```
exit &main();
```

```
sub main {
```

```
    my $oCGI = CGI->new();
```

```
    my $sEmailAddress = $oCGI->param('txtEmailAddress');
    my $sUsername      = $oCGI->param('txtUsername');
```

```
    my $oContent      = new XDrive::Template( {'partner_code' => 'xdrv'} );
    my $oNavigation    = new XDrive::Template( {'partner_code' => 'xdrv'} );
    my $oLayout        = new XDrive::Template( {'partner_code' => 'xdrv'} );
```

```
    ## Load the required template HTML files.
```

```
    $oNavigation->load("front_nav.html");
```

```
    $oContent->load("front_signup.html");
```

```
    $oLayout->load("layout.html");
```

```
    if ( ($sEmailAddress) && ($sUsername) ) {
```

```
        ## Change user's password
```

```
        my @characters = ('a'..'z','A'..'Z','0'..'9');
```

```
        my $sRandomKey = XDRandomString(8,\@characters);
```

```
        if(&PasswordSet($oContent,$sUsername, $sEmailAddress, $sRandomKey)) {
            sendMail($oContent,$sUsername, $sRandomKey, $email_template);
        }
```

```
        &display_form($oContent,$thank_you_template);
```

```
    } else {
```

```
        &display_form($oContent,$request_template);
```



```

    }

    ## Print out the HTML and exit
    $oLayout->tags
        (
            'header_graphic' => 'header_fill.gif',
            'title' => 'What is my password?',
            'content' => $oContent->get,
            'navigation' => $oNavigation->get,
        );
    $oLayout->clear;

    print header,$oLayout->get;
    return 0;
}

sub PasswordSet
{
    my($oContent,$sUsername, $sEmailAddress, $sPassword) = @_ ;
    my $bReturnValue = 0;
    my $status;
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new();
    my $oUser = XDrive::DatabaseO::Table::UserData->new(undef, $oDiskAccount-
>fetchDBO());

    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oUser->loadWhere("SEQ", $oDiskAccount->fetchColumn("USER_SEQ"));

    if ( (defined $oDiskAccount->fetchColumn("USER_SEQ"))
        &&($oUser->fetchColumn("EMAIL_ADDRESS") eq $sEmailAddress)
        )
    {
        my $sPassEncrypted = XDEncrypt($sPassword);
        $oDiskAccount->setColumn("PASSWORD", $sPassEncrypted);
        $oDiskAccount->update();
        $oDiskAccount->commit();
        $bReturnValue = 1;
    }
    elsif( (defined $oDiskAccount->fetchColumn("USER_SEQ"))
        &&($oUser->fetchColumn("EMAIL_ADDRESS") ne $sEmailAddress)
        )
    {
        &sendMail($oContent,$sUsername, "", $alert_template, " NOT");
    }

    $oDiskAccount->finish();
    $oDiskAccount->disconnect();

    return $bReturnValue;
}

sub display_form {
    my ($oContent,$template) = @_ ;
    $oContent->load($template);
}

sub sendMail {
    my ($oContent,$username, $password, $template, $not) = @_ ;

    my ($name_first, $name_last, $email_address, $data);
    my $oSearch = XDrive::DatabaseO::Search->new(undef);

```

```

$data = $oSearch->XDUserInfoByUsername($username);
$name_first = $data->[0]->[0];
$name_last = $data->[0]->[1];
$email_address = $data->[0]->[2];
$username = $data->[0]->[3];

my $message = &get_message($oContent,$name_first, $name_last, $username,
$password, $template);

my %toXdrive =
(
    To      => "$name_first $name_last <$email_address>",
    Bcc     => '',
    From    => "support@xdrive.com",
    Message => $message,
    Subject => "X:drive Password$not Updated!"
);

sendmail(%toXdrive);
}

sub get_message {
my ($oContent,$name_first, $name_last, $username, $password, $template) =
@_;

$name_first = ($name_first)? $name_first : "";
$name_last = ($name_last)? $name_last : "";

$oContent->load($template);
$oContent->tags
(
    (
        'name_first' => $name_first,
        'name_last'  => $name_last,
        'password'   => $password,
        'username'   => $username,
    )
);

return $oContent->get;
}

```

###forgot_username.cgi

```
#!/usr/bin/perl

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header param);
use CGI::Carp qw(fatalsToBrowser);
use Mail::Sendmail;

use Token;
# use XDrive::CGI qw(:MAIN);
use XDrive::Template;
use XDrive::DatabaseO::Search;
use XDrive::Library;
use XDrive::Utils::RandomString;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;

#####
my $invalid_template = "invalid_email.thtml";
my $request_template = "forgot_username_request.thtml";
my $thank_you_template = "forgot_username_t_y.thtml";
my $email_template = "forgot_username_email.thtml";
#####

exit &main();

sub main {
    my $oCGI = CGI->new();

    my $sEmailAddress = $oCGI->param('txtEmailAddress');
    my ($ar_usernames, $length);

    my $oSearch = XDrive::DatabaseO::Search->new(undef);

    my $oContent = new XDrive::Template;
    my $oNavigation = new XDrive::Template;
    my $oLayout = new XDrive::Template;

    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');
    $oLayout->partner('xdrv');

    ## Load the required template HTML files.
    $oNavigation->load("front_nav.thtml");
    $oLayout->load("layout.thtml");

    ## IF a parameter of email adress has been processed
    ## and in the correct format, then retrieve usernames
    ## associated with the email and send them.

    if ($sEmailAddress)
    {
        ## * * * * *
        ## added by kanlaya to check for correct email format
        ## * * * * *

        if ($sEmailAddress =~ /.*\@.*\./)

```

```

    {

        ## Takes the email_address and returns an array_ref
        ## of all the disk_account.usernames associated
        ## with that users user.email_address
        $ar_usernames = $oSearch->XDUsernameFromEmail($sEmailAddress);
        $length = @$ar_usernames;

        ## IF there are usernames found for this address,
        ## then email the address the list of usernames.
        if($length > 0)
        {
            &sendMail($ar_usernames, $sEmailAddress, $length);
        }

        $oContent->load($thank_you_template);
        $oContent->tags({'emailAddress' => $sEmailAddress,});

    }

    else
        ($oContent->load($invalid_template));
}
## * * * * *
## end add
## * * * * *
}
else
    ($oContent->load($request_template));

## Print out the HTML and exit
$oLayout->tags
(
    'header_graphic' => 'header_fill.gif',
    'title' => 'What is my username?',
    'content' => $oContent->get,
    'navigation' => $oNavigation->get,
);
$oLayout->clear;

print header, $oLayout->get;

return 0;
}

sub sendMail {
    my ($usernames, $email, $length) = @_;

    my $message = &get_message($usernames, $email, $length);

    my %toXdrive =
    (
        To      => "$email",
        Bcc     => '',
        From    => "support@xdrive.com",
        Message => $message,
        Subject => "X:drive Username Reminder"
    );

    sendmail(%toXdrive);
}

sub get_message {
    my ($usernames, $email, $length) = @_;

```

```
my ($sUsername, $sPluralS, $sPluralVerb);
$sUsername = join("\n", @$usernames);
$sPluralS = ($length > 1)? "s" : "";
$sPluralVerb = ($length > 1)? "are" : "is";

my $oForm = new XDrive::Template;
$oForm->partner('xdrv');

$oForm->load($email_template);
$oForm->tags
(
    'sEmailAddress' => $email,
    'sUsername' => $sUsername,
    'sPluralS' => $sPluralS,
    'sPluralVerb' => $sPluralVerb
);
$oForm->clear;

return $oForm->get;
}
```

###frame_generic.cgi

```

#!/usr/bin/perl
## Written by Matt Clapp on 6/28/99
## This CGI allows us to pass the sst and sid on to the inner frame

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use Token;
use XDrive::Library;
use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::Client::Security;
use XDrive::CGI qw(XDErrorToBrowser);
use XDrive::CGI::Cookie;

&main;
exit;

sub main
{
    my $oCGI = CGI->new();

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $language = $oCookie->getElement('language');
    $language = 'english' unless $language;

    my $sThtmlFile = $oCGI->param('thtml');
    my $sFrameHeight = $oCGI->param('sFrameHeight');

    if ($sFrameHeight == "")
    {
        $sFrameHeight="40";
    }

    if ($sThtmlFile eq 'download_client.thtml')
    {
        my $oTemplate = new XDrive::Template( { 'partner_code' => 'xdrv' } );
        $oTemplate->load($sThtmlFile);
        $oTemplate->tags( { 'sFrameHeight' => $sFrameHeight,
                           'language' => $language } );
        print "Content-type: text/html\n\n";
        print $oTemplate->get();
    }
    elsif ($sThtmlFile eq 'centerview.thtml')
    {
        my $sFrameSet;
        if ($sFrameHeight > 1)
        {
            $sFrameSet = "$sFrameHeight,*";
        }
        else
        {
            $sFrameSet = "100%,*";
        }
    }
};

```

```

        print <<EOM;
Content-type: text/html

```

```

<FRAMESET ROWS="$sFrameSet" BORDER=0 FRAMEBORDER=0 MARGINWIDTH=0
MARGINHEIGHT=0 TOPMARGIN=0 LEFTMARGIN=0 frameBorder=0 frameSpacing=0>
EOM

```

```

        if ($sFrameHeight > 1)
        {
            print <<EOM;
            <FRAME NAME='controls' SRC='/explorer/$language/buttons.html'
            SCROLLING=NO MARGINWIDTH=0 MARGINHEIGHT=0 TOPMARGIN=0 LEFTMARGIN=0>
            EOM
        }

        print <<EOM;
        <FRAME NAME='userData' SRC='/cgi-bin/explorer_user_data.cgi'
        SCROLLING=AUTO MARGINWIDTH=0 MARGINHEIGHT=0 TOPMARGIN=0 LEFTMARGIN=0>
        </FRAMESET>
        EOM
    }
    else
    {
        ## Security check. Since the thtml file is passed in via the URL
the server
        ## can be hacked by passing in ../ offsets to get the directory
the hacker
        ## wants. A cleaner way would be to pass in a number and use
that number
        ## to access a hash, and die with a security violation if no such
has key
        ## exists.

        my $oDBO = XDrive::DatabaseO->new(undef,undef);
        my $oErr = new XDrive::Error;
        my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

        ####
        ## If the user failed to authenticate or an error occurred then
        ## redirect them to the error CGI and exit
        ####

        if ($oErr->Occurud)
        {
            xd_fatal_error($oCGI,$oErr);
            $oDBO->disconnect();
        }

        warn "#ALERT hacking attempt by ".$oToken->data('user').
            " from ".$ENV{REMOTE_IP};
        my $sMessage = $oErr->ReturnMessageGivenCode(341);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
        $oDBO->disconnect();
        exit;
    }
}

```

###get_a_shared_file.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);

use XDrive::CGI;
use XDrive::Template;
use XDrive::DatabaseO::Table::DiskItemShare;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::Reseller;
use XDrive::CGI::Cookie;

use strict;

exit &main();

sub main {
    my $cgi = CGI->new();
    my ($ClaimTicket, $oPage, $xdDBH);

    if ($ENV{'QUERY_STRING'} !~ /=/)
    {
        $ClaimTicket = $ENV{'QUERY_STRING'};
    }
    else
    {
        $ClaimTicket = $cgi->param("claim_ticket");
    }

    if (length($ClaimTicket) < 5)
    {
        $ClaimTicket = $ENV{'PATH_INFO'};
        $ClaimTicket =~ s/^\///;
    }

    ##make sure that if claim ticket ends in -SP we set language to spanish
and
    ##truncate claim ticket
    if ($ClaimTicket =~ /-SP$/)
    {
        $ClaimTicket = substr($ClaimTicket,0,length($ClaimTicket)-3);
        my $oCookie = new XDrive::CGI::Cookie('x_session_info', $cgi);
        $oCookie->setElement
        (
            {
                'language' => 'spanish',
            }
        );

        print "Set-Cookie: ". $oCookie->asString();
    }

    my $oDiskItemShare = XDrive::DatabaseO::Table::DiskItemShare->new();
    $oDiskItemShare->loadWhere("RANDOM_KEY", $ClaimTicket);

    my $diskAccount = $oDiskItemShare-
>fetchColumn("DISK_ACCOUNT_USER_SEQ");
    $xdDBH = $oDiskItemShare->fetchDBO();
}
```



```

my $oUserAccount = XDrive::DatabaseO::Table::UserData->new(undef,
$xdDBH);
    $oUserAccount->loadWhere("SEQ", $diskAccount);

    my $oReseller = XDrive::DatabaseO::Table::Reseller->new(undef,
$xdDBH);
    $oReseller->loadWhere("SEQ", $oUserAccount-
>fetchColumn("RESELLER_SEQ"));

    my $spartner = $oReseller->fetchColumn("CODE");

    ## If the disk item share was not in the database then just use an
xdrive
    ## look n' feel. NOTE!!!!!! This should be changed to a plain looking
    ## error screen.
    $spartner = 'xdrv' if ! defined $spartner;

    $oPage = new XDrive::Template
        (
            'partner_code' => $spartner
        );

    $oPage->load('get_a_shared_file__frameset.shtml');
    $oPage->tags
        (
            'ClaimTicket' => $ClaimTicket,
            'referee' => $diskAccount,
        );
    $oPage->clear();
    print $cgi->header, $oPage->get;

    $oDiskItemShare->disconnect();
    return 0;
}

```

###get_a_shared_file_download.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use Data::Dumper;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::DatabaseO::Search;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::DiskItemShare;
use XDrive::Template;
use XDrive::Error;

use strict;

&main;
exit;

sub main
{
    my ($sFileDescription, $sFileSize, $sRandomKey, $sSeq);

    my $cgi = CGI->new();
    my $oErr = new XDrive::Error;
    my $g_oShared;    ## Shared object
    my $g_oSearch;    ## Shared object
    my $g_oAction;    ## Action object
    my $g_oFileStat;  ## File stats
    $sRandomKey = $ENV{'QUERY_STRING'};

    if (!$sRandomKey)
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1360);
        &display_error($sMessage,$oErr);
    }
    else
    {
        ## Instantiate and load the shared object.
        $g_oShared = XDrive::DatabaseO::Table::DiskItemShare->new(undef,
undef);
        $g_oSearch = XDrive::DatabaseO::Search->new($g_oShared->
>fetchDBO());

        $g_oShared->loadWhere("RANDOM_KEY", $sRandomKey);
        $sSeq = $g_oShared->fetchColumn("SEQ");

        if (!$sSeq)
        {
            my $sMessage = $oErr->ReturnMessageGivenCode(1361);
            &display_error($sMessage,$oErr);
        }

        ## Call the client action constructor with the shared object
        ## which it will use to load all the needed client information.
        $g_oAction = new XDrive::Client::Actions($g_oShared,$cgi);

        my $sFile = join
            ('/',
```

```

        $g_oShared->fetchColumn("ITEM_PATH"),
        $g_oShared->fetchColumn("ITEM_NAME")
    );
    $g_oFileStat = $g_oAction->FileStat($sFile);

    if (!$g_oFileStat) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1362);
        &display_error($sMessage,$oErr);
    } else {

        $sFileDescription = $g_oShared->fetchColumn("DESCRIPTION");
        $sFileSize = ($g_oFileStat->size() > 1024)?
int($g_oFileStat->size()/1024) . "K" :
        $g_oFileStat->size() . " bytes";

        &display_form($g_oShared-
>fetchColumn("ITEM_NAME"),$sRandomKey, $sFileSize,
$sFileDescription,$g_oSearch->XDResellerCodeFromUserSeq($g_oShared-
>fetchColumn("DISK_ACCOUNT_USER_SEQ")));
    }
    $g_oShared->finish();
    $g_oShared->disconnect();
    $g_oAction->DisconnectDB();
}

```

```

sub display_form
{
    my ($sFilename,$sRandomKey, $sFileSize, $sFileDescription,$sPartner)
=@_;

```

```

    my $oForm = new XDrive::Template;
    $oForm->partner($sPartner);
    $oForm->load('get_a_shared_file__download_screen.shtml');
    $oForm->tags
    (
        {
            'sFilename' => $sFilename,
            'sExtraPathInfo' => $sFilename,
            'sRandomKey' => $sRandomKey,
            'sFileSize' => $sFileSize,
            'sFileDescription' => $sFileDescription,
        }
    );

```

```

    $oForm->clear();
    print "content-type: text/html\n\n", $oForm->get;
    exit(0);
}

```

```

sub display_error
{

```

```

    my ($message,$oErr) = @_;

    if (!$message)
    {
        $message = $oErr->ReturnMessageGivenCode(1363);
    }

```

```

    my $oForm = new XDrive::Template;
    $oForm->partner('xdrv');
    $oForm->load('get_a_shared_file__error.shtml');
}

```

```
$oForm->tags
((
    'message' => $message,
));
print "content-type: text/html\n\n", $oForm->get;
exit(0);
}
```

###login.cgi

```
#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> to verify that the user is
# good to login, if they are then log them in and otherwise redirect to
# a not authorized page.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::UserQuota;
use XDrive::DatabaseO::Table::Language;
use XDrive::DatabaseO::Search;
use CGI qw(param redirect header cookie);
use CGI;
use XDrive::CGI::Cookie;
use CGI::Carp qw(fatalsToBrowser);
```

```
use XDrive::CGI;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Template;
use XDrive::Library;
use XDrive::DatabaseO;
use Mail::Sendmail;
```

```
&main;
exit;
```

```
sub main
{
    my $oCGI      = new CGI;
    my $oErr      = new XDrive::Error;
    ##my $oDBO    = new XDrive::DatabaseO;
    my $oCookie   = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    my $oToken;
    my $sToken;
    my $sUsername;
    my $sPartnerCode;
    ## johngaa add for dbexist check
    my $oDBO;

    if (XDDDBConnectionCheck() && XDNFSCheck())
    {
        $oDBO = new XDrive::DatabaseO;
    }
    else
    {
        $oDBO = undef;
        print redirect("/upgrading_index.html");
        exit;
    }
    ## end of johngaa change
    my $bSecurity = $oCGI->param('bSecurity');
    my $sPartnerToken = $oCGI->param('partner_token');

    my $passed_lang = $oCGI->param('language');

    #####
```

```

## Attempt to authenticate the user by using one of the following two
## authentication methods: username/password pair or partner token
## authentication.
#####
if (! defined $sUsername && length($sPartnerToken) > 20)
{
    authPartnerUser($oCGI,$oErr,$oDBO,\$sUsername,\$oToken,
        \$sPartnerCode,$sPartnerToken);
    $sToken = $oToken->name();
}
else
{
    authWebSiteUser($oCGI,$oErr,$oDBO,\$sUsername,\$oToken);
    $sPartnerCode = 'xdrv';
}

#####
## If an error occurred while trying to create a token then redirect
## the user to the error page.
#####
if ($oErr->Occurud)
{
    $oDBO->disconnect;
    xd_fatal_error($oCGI,$oErr);
    exit;
}

#####
## If we have gotten here then we have an authenticated user.
#####

#####
## Build and print out cookies
#####
my $sLanguage = getLanguage($oDBO,$sUsername);

##check if user's language is the same as passed language
if ((length($passed_lang) > 0) && $sLanguage ne $passed_lang)
{
    ##update db here to new language
    setLanguage($oDBO,$sUsername,$passed_lang);
    ##update session to new language
    $sLanguage = $passed_lang;
}

##delete the promo cookie; this will not be set here and we
##don't want an old one hanging out
##promo cookies should be set in promo.cgi
$soCookie->deleteElement('promo') if $soCookie->getElement('promo');

$soCookie->setElement
(
    (
        'language' => $sLanguage,
        'partner' => $sPartnerCode,
    ));

print "Set-Cookie: ". $soCookie->asString();
print "Set-Cookie: SST=$sToken; domain=xdrive.com; path=/\n"
    if $sPartnerCode ne 'xdrv';

```

```
#####
## write user login to the database
#####
&incrementLoginNumber($oDBO,$sUsername,$sLanguage,$sPartnerCode);
```

```
#####
## Send the user off into thier file explorer
#####
if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
{
    print $oCGI->redirect("?sst=". $oToken->name(). "&sid=0");
}
else
{
    xd_web_open($oCGI, "", "", \%ENV, $bSecurity);
}

$oDBO->disconnect;
return 0;
}
```

```
sub isYesterday()
{
##
## Date: 01/25/99
## used to check of a date if its today or not
##
```

```
    my $last_login = `shift`;
    my $nSec;                ## Seconds
    my $nMin;                ## Minutes
    my $nHour;               ## Hours
    my $sDay;                ## Weekday
    my $nDay;                ## Numeric date (01-31)
    my $nMonth;              ## Numeric month (01-12)
    my $nYear;               ## Numeric year (00-99)
```

```
    my $todaysDate = ($nSec, $nMin, $nHour, $nDay, $nMonth, $nYear,
    $sDay) = (localtime(time))[0,1,2,3,4,5,6];
```

```
    $last_login =~ /([\d]+)-([\d]+)-([\d]+)/i;
    my $last_login_year = int($1);
    my $last_login_month = int($2);
    my $last_login_day = int($3);
```

```
    if ($last_login_year < $nYear)
    {
        return 1;
    }
    if ($last_login_month < $nMonth)
    {
        return 1;
    }
    if ($last_login_day < $nDay)
    {
        return 1;
    }
    return 0;
```

```
}
```

```

sub incrementLoginNumber()
{
    my $oDBO = shift;
    my $sUsername = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $timesLoggedIn = $oDiskAccount->fetchColumn("LOGIN_NUM");
    my $user_seq = $oDiskAccount->fetchColumn("USER_SEQ");

    ## johngaa add
    ## insert a warn in the error log if this is the
    if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
    {
        my $todaysDate = XDToday();
        warn "#client_login user_seq=$user_seq username=$sUsername
date=$todaysDate#";
    }
    ## end of johngaa warn of first entry

    if ($timesLoggedIn)
    {
        $timesLoggedIn++;
    }
    else
    {
        $timesLoggedIn=1;
    }

    $oDiskAccount->setColumn("LOGIN_NUM", $timesLoggedIn);
    $oDiskAccount->setColumn("LAST_LOGIN",XDToday());

    my $status = $oDiskAccount->update();

    if ($status > -1)
    {
        $oDiskAccount->commit();
        $oDiskAccount->finish();

        ## johngaa modify to exclude college club
        ## and quepasa users out of the extra space
        ## promo

        my $oUserData = XDrive::DatabaseO::Table::UserData-
>new(undef,$oDBO);
        $oUserData->loadWhere("SEQ", $user_seq);
        my $reseller_seq = $oUserData-
>fetchColumn("RESELLER_SEQ");
        if (!(isResellerSeqCC_QUPA($oDBO,$reseller_seq)))
        {
            ##give user extra 10MB if 10th login
            if ($timesLoggedIn == 10)
            {
                my $oUserQuota = XDrive::DatabaseO::Table::UserQuota-
>new(undef,$oDBO);

```



```

        $oUserQuota->loadWhere("USER_SEQ", $user_seq);
        my $additional_quota = $oUserQuota-
>incrementQuota($user_seq,10240);
        if ($additional_quota > 0)
        {
            &send_email($user_seq, $oDBO,
$additional_quota,$sLanguage, $sPartnerCode);
        }
    }
}
else
{
    # $oDiskAccount->rollback();
}
}

```

```

sub isResellerSeqCC_QUPA
{
    my $oDBO = shift;
    my $reseller_seq = shift;
    my $dbh = $oDBO->fetchDBH();

    my $sql_stmt = "SELECT code FROM reseller WHERE seq=?";
    my $cmd;
    my @data;

    $cmd = $dbh->prepare($sql_stmt);
    $cmd->execute(($reseller_seq));
    @data = $cmd->fetchrow_array;
    if ($data[0] eq 'cc' || $data[0] eq 'qupa')
    {
        return 1;
        ##print "should return a true\n"
    }
    return 0;
}

```

```

sub send_email
{
    my $user_seq = shift;
    my $oDBO = shift;
    my $additional_quota = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    if ($sPartnerCode eq 'cc')
    {
        return;
    }

    ##comes in as k, change to megabytes
    my $mbs = $additional_quota/1024;

    my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef,$oDBO);
    $oUserData->loadWhere("SEQ", $user_seq);
    my $email_address = $oUserData->fetchColumn("EMAIL_ADDRESS");
    my $name_first = $oUserData->fetchColumn("NAME_FIRST");
    my $name_last = $oUserData->fetchColumn("NAME_LAST");
}

```

```

    my $oTemplate = new XDrive::Template
    (
        'language' => $sLanguage,
        'partner_code' => $sPartnerCode,
    );

    $oTemplate->load('received_10MB_10logins.thtml');
    $oTemplate->tags(
        'mbs' => $mbs,
    );
    $oTemplate->clear();
    my $message = $oTemplate->get;

    my %toXdrive =
    (
        To      => "$name_first $name_last <$email_address>",
        Bcc      => '',
        From     => "support@xdrive.com",
        Message  => $message,
        Subject  => "Congratulations!"
    );

    sendmail(%toXdrive);
}

sub authPartnerUser
{
    my $oCGI = shift;
    my $oErr = shift;
    my $oDBO = shift;
    my $rsUsername = shift;
    my $roToken = shift;
    my $rsPartnerCode = shift;
    my $sPartnerToken = shift;

    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    my $oPartnerToken = new Token
    (
        'err' => $oErr,
        'dbh' => $oDBO,
    );
    $oPartnerToken->load($sPartnerToken);

    return if $oErr->Occurud;

    $$roToken = new Token
    (
        'dbh' => $oDBO,
        'err' => $oErr,
        'user_sequence' => $oPartnerToken->data('user_seq'),
    );
    $$roToken->create();

    return if $oErr->Occurud;

    ### Edited by Justin so that the partner_code is looked for
    ### in the cookie instead of the token table.
    ### And then again because I shouldn't have done that. The
    ### partner code hasn't been set in the cookie by this point,
    ### so we shouldn't be looking in there for it.
    $$rsPartnerCode = $oPartnerToken->data('partner_code');
    # $$rsPartnerCode = $oCookie->getElement('partner');

```

```

    $$rsUsername = $oPartnerToken->data('user');

    $$roToken->data('ip', $ENV{REMOTE_ADDR});
    $$roToken->data('browser', $ENV{HTTP_USER_AGENT});
    $$roToken->data('user', $$rsUsername);
    $$roToken->data('user_seq', $oPartnerToken->data('user_seq'));
    $$roToken->data('partner_code', $$rsPartnerCode);
    $$roToken->data('disk_account_seq', $oPartnerToken-
>data('disk_account_seq'));
    $$roToken->save;

    $oPartnerToken->delete();
}

sub authWebSiteUser
{
    my $oCGI = shift;
    my $oErr = shift;
    my $oDBO = shift;
    my $rsUsername = shift;
    my $roToken = shift;

    my $sPassword = $oCGI->param('pass');
    $$rsUsername = $oCGI->param('user');

    if (xd_auth_password($$rsUsername, $sPassword, $oDBO))
    {
        ## Login the user info X:drive and get the session token
        $roToken = xd_login($oCGI, $$rsUsername, $oErr, $oDBO);
    }
    else
    {
        $oErr->AddErrorByErrorCode('501');
    }
}

sub getLanguage
{
    my $oDBO = shift;
    my $sUsername = shift;

    my $language;

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, $oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings-
>new(undef, $oDBO);
    $oUserSettings->loadWhere("USER_SEQ", $userSeq);
    $oUserSettings->finish;
    my $language = $oUserSettings->fetchColumn("LANGUAGE");

    if ($language eq '')
    {
        $language = 'english';
    }
    else
    {
        ## Get language from database given code

```

```

        my $oLanguage = XDrive::DatabaseO::Table::Language-
>new(undef,$oDBO);
        $oLanguage->loadWhere("SEQ",$language);
        $oLanguage->finish;
        $language = $oLanguage->fetchColumn("CODE");
    }

    return $language;
}

sub setLanguage
{
    ##set the LANGUAGE column of the User_Settings table to passed
    language

    my $oDBO = shift;
    my $sUsername = shift;
    my $language = shift;

    my ($rv,$errorCode);

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBO);
    ##grab right table
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings-
>new(undef,$oDBO);
    $oUserSettings->loadWhere("USER_SEQ",$userSeq);
    $oUserSettings->finish;

    ##grab the seq number of the LANGUAGE being passed
    my $oLanguage = XDrive::DatabaseO::Table::Language->new(undef,$oDBO);
    $oLanguage->loadWhere("CODE",$language);
    $oLanguage->finish();
    my $seq_lang = $oLanguage->fetchColumn("SEQ");

    eval
    {
        ##
        ##set language here
        $rv = 0;
        $oUserSettings->setColumn('LANGUAGE',$seq_lang);
        $rv = $oUserSettings->update();
    };
    if ($rv == 0)
    {
        $oUserSettings->rollback();
        $errorCode = 0;
    }
    else
    {
        $oUserSettings->commit();
        $errorCode = 1;
    }
    return $errorCode;
}

```

###logout.cgi

```
#!/usr/bin/perl
## Program to log the user out, currently hacked to redirect to the homepage.
## Modified by Justin on 10/15/99 to be Security.pm friendly
## and get rid of the XDrive::CGI stuff.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::Error;

&main;
exit;

sub main
{
    my $oCGI    = CGI->new();
    my $oDBO    = new XDrive::DatabaseO;
    my $oError  = new XDrive::Error;

    ##removes token from the database
    xd_logout($oDBO, $oCGI, $oError);

    $oDBO->disconnect;

    print $oCGI->redirect('/');
    return 0;
}
```

###navbar.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <martin@xdrive.com> on Sun Sep 5 1999
## Script to dynamically show the correct tempate based on which
## partner is looking at the web site.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use XDrive::Library;
use XDrive::Template;
use XDrive::Error;
use XDrive::DatabaseO;
use XDrive::Client::Security;

&main;
exit;

sub main
{
    ## Load the session token
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;
    my $oCGI = new CGI;
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    #####
    ## Attempt to autenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If the user does not validate or an error occurred
    ## then redirect to the error CGI and exit
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);

        $oDBO->disconnect();

        exit;
    }

    #####
    ## Otherwise we have validated and should load the navbar
    ## associated with the partner
    #####
    ## Edited by Justin so that partner_code is looked for in
    ## the cookie instead of the token table.
    # my $oForm = new XDrive::Template
    #   ({
    #     'partner_code' => $oToken->data('partner_code')
    #   });

    my $oForm = new XDrive::Template
      ({
```

```
'partner_code' => $oCookie->getElement('partner')
));

$oForm->load('navbar.html');

####
## Print the navbar and stop
####

print "Content-type: text/html\n\n";
print $oForm->get;

$oDBO->disconnect();

return 0;
}
```

###password_change.cgi

```

#!/usr/bin/perl
## Written by Lucas McGregor on ???

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header param);
use CGI::Carp qw(fatalsToBrowser);

use Token;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::Client::Registration;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::Library;
use XDrive::Template;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;

&main;
exit;

sub main
{
    my $oCGI    = CGI->new();
    my $oDBO    = new XDrive::DatabaseO;
    my $oErr    = new XDrive::Error;

    ####
    ## Attempt to autenticate the user
    ####

    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    ####
    ## If an error occurs or the user fails to autenticate then redirect
    ## to the error CGI and exit
    ####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    ####
    ## Otherwise have have autenticated the user and can proceed
    ####

    my $sUsername = $oToken->data('user');

    my $sPasswordNew      = $oCGI->param('txtPasswordNew1');
    my $sPasswordNewConfirm = $oCGI->param('txtPasswordNew2');
    my $sPasswordOld      = $oCGI->param('txtPasswordOld1');

```



```

    if (($sPasswordNew eq '') || ($sPasswordNewConfirm eq '') ||
($sPasswordOld eq ''))
    {
        ##if any of the fields is blank, give em error message
        my $sMessage = $oErr->ReturnMessageGivenCode(1340);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    ## Change user's password
    PasswordSet($sUsername, $sPasswordNew,
$sPasswordOld, $oToken, $oErr, $oCGI);

    return 0;
}

#####
## PasswordSet: Change user's password
#####

sub PasswordSet($$)
{
    my $sUsername = shift;          ## (I) User in question
    my $sPassword = shift;          ## (I) New password
    my $sPasswordOld = shift;        ## (I) Old password
    my $oToken = shift;              ## (I) Token object
    my $oErr = shift;
    my $oCGI = shift;
    my $sPassEncrypted = XDEncrypt($sPassword);

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new();
    $oDiskAccount->loadWhere("USERNAME", $sUsername);

    if (! PasswordsMatch($oDiskAccount->fetchColumn("PASSWORD"), $sPasswordOld))
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1341);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    if (! defined $oDiskAccount->fetchColumn("USER_SEQ"))
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1342);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    $oDiskAccount->setColumn("PASSWORD", $sPassEncrypted);
    my $status = $oDiskAccount->update();

    ## If no error, then commit
    ## Else rollback and show an error
    if ($status > -1) {
        $oDiskAccount->commit();
    }
    else
    {
        $oDiskAccount->rollback();
        my $sMessage = $oErr->ReturnMessageGivenCode(1343);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    $oDiskAccount->finish();
    $oDiskAccount->disconnect();
}

```

```

my $oTemplate = new XDrive::Template( {'partner_code' => 'xdrv'} );
$oTemplate->load('password_changed.thtml');
print $oCGI->header(), $oTemplate->get;
}

#####
## PasswordsMatch: Check an encrypted password against an unencrypted
## password and return true or false.
#####

sub PasswordsMatch
{
    my $sEncrypted = shift; ## current password
    my $sToCheck   = shift; ## string to check

    ## Encrypt the passed password with the salt from the password taken
    ## from the database.
    my ($sSalt) = $sEncrypted =~ /\^(\\w{2})/;

    ## Do the passwords match? If so then return true, otherwise false.
    if ($sEncrypted eq crypt($sToCheck,$sSalt))
    {
        return TRUE;
    }

    return FALSE;
}

```

###promo.cgi

```

#!/usr/bin/perl
##
## File: promo.cgi
##
## Written by Justin White on 10/25/99.
## Sets a promo cookie and redirects to the home page.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::CGI::Cookie;
use XDrive::DatabaseO::Search;

use CGI;
use CGI::Carp qw(fatalsToBrowser);

&main();

exit;

sub main {
    my ($cookie, $promo, $new_info, $oSearch, $oTemplate);

    my $oCGI = CGI->new();
    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    my $path_info = $ENV{'PATH_INFO'};

    my $sClaimTicket = $oCGI->param('ct');

    if ($sClaimTicket) {
        ##
        # Via cookie, set the promo so that signup_account.cgi treats
        # it as a promo and set the claim ticket code so that we can
        # remove that data from the batch_user_data table.
        ##
        $oCookie->setElement( {'ct' => $sClaimTicket} );
    }

    if ($path_info) {
        $path_info =~ s/^\///;

        $oCookie->setElement( {'promo' => $path_info} );
        $oCookie->setPath('/');

        ##if user is coming from the befree promo, set a cookie with their
        ##source id, be Free requires this for tracking purposes
        if ($path_info =~ /befree/)
        {
            my $sourceid = $oCGI->param('sourceid');
            print "Set-Cookie: sourceid=$sourceid; domain=.xdrive.com;

path=/\n"
        )

        my $oDBO = XDrive::DatabaseO->new();
        my $oSearch = XDrive::DatabaseO::Search->new($oDBO);

```

```

my @bind_array = ($path_info);

# my $st = "SELECT p.template, p.redirect_url, dl.code
#         FROM xdrive.promo p, xdrive.v_language dl
#         WHERE p.uri = '$path_info'
#         AND p.du_language = dl.seq(+)";

my $st = "SELECT p.template, p.redirect_url, dl.code
        FROM xdrive.promo p, xdrive.v_language dl
        WHERE p.uri = ?
        AND p.du_language = dl.seq(+)";

# my $data = $oSearch->XDSQLSearch($st);
my $data = $oSearch->XDSQLSearch($st, \@bind_array);
my $rows = @{$data};

if ($rows > 0) {
    my $template      = $$data[0][0];
    my $redirect_url  = $$data[0][1];
    my $language      = $$data[0][3];

    $oCookie->setElement( {'language' => $language} );

    print "Set-Cookie: ", $oCookie->asString();

    if ($template) {
        eval {
            $oTemplate = new XDrive::Template( {'cookie'      =>
$oCookie,
                                                'partner_code' => 'xdrv' }
);
            $oTemplate->partner('xdrv');

            $oTemplate->load("promo/$template");
        };

        if ($?) {
            print $oCGI->redirect('/');
            warn "$?\n";
        }
        else {
            print $oCGI->header(), $oTemplate->get;
        }

        $oSearch->disconnect;
    }
    elsif ($redirect_url) {
        print $oCGI->redirect($redirect_url);

        $oSearch->disconnect;
    }
    else {
        print $oCGI->redirect('/');

        $oSearch->disconnect;
    }
}
else {
    print $oCGI->redirect('/');
}
}
else {

```

WO 01/33381

PCT/US00/30536

```
    print $oCGI->redirect('/');  
  
    $oSearch->disconnect;  
}  
  
return;  
}
```

###removespace.cgi

```

#!/usr/bin/perl
#####3
## Written by Karen Eppinger
## removespace.cgi - cancels additional space requests
#####3

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Error;
use XDrive::Library;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::Reseller;
use XDrive::DatabaseO::Table::Deal;
use XDrive::DatabaseO::Table::Item;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::UserPurchase;
use XDrive::Client::Actions;
use XDrive::DatabaseO::Search;
use XDrive::Sale::Purchase;
use Mail::Sendmail;
use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Template;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use EpochClient_ssl;

use strict;

$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)); # Make $ENV safer

exit &main;

#####
## main: main function calls all others
##
##
#####

sub main
{
    my $oCGI = CGI->new();
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    ####
    ## Attempt to authenticate the user
    ####

    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    ####
    ## If an error occurs during authentication or authentication fails
    ## then redirect to the error CGI and exit
    ####

```

```

if ($oErr->Occurud)
{
    xd_fatal_error($oCGI,$oErr);
    exit;
}

####
## Otherwise we have a valid session
####

my $sUserName = $oToken->data('user');
my $oTemplate = new XDrive::Template
(
    (
        'partner_code' => $oToken->data('partner_code')
    )
);

## used to figure whether to give user the form or process the form
my $sAction = $oCGI->param("action");

## Create a DBH
my $oDBH = XDrive::DatabaseO->new();

## if the action is a request type
if ($sAction eq 'process')
{
    ##else we process the form input
    &CheckSpaceUsed($oCGI,$sUserName,$oTemplate,$oToken,$oDBH,$oErr);
}
elsif ($sAction eq 'intro')
{
    &ShowIntroPage($oTemplate,$sUserName,$oToken,$oCGI);
}
else
{
    ## we give the user the form
    &ShowSpace($sUserName,$oTemplate,$oToken,$oDBH,$oErr);
}
$oDBH->disconnect();
}

```

```

#####
## CheckSpaceUsed: make sure the user has enough free space for his files
## if not, do not let him cancel
#####

```

```

sub CheckSpaceUsed

```

```

{
    my $oCGI = shift;
    my $sUserName = shift;
    my $oTemplate = shift;
    my $oToken = shift;
    my $oDBH = shift;
    my $oErr = shift;

    ##we need to get the number of fields so we know what to process
    my @fields = $oCGI->param;
    my $checked = 0;

    my $returnValue = '';
    ##for each checked item, either cancel or tell user they may not cancel
    ##because space used is larger than space available after cancelation

```

```

for (my $i=0; $i<$#fields; $i++)
{
    if ($fields[$i] =~ /^tc_/)
    {
        $fields[$i] =~ s/^tc_//;
        my $oPurchase = new XDrive::Sale::Purchase($oDBH);
        my @message_dbmessage = $oPurchase->CancelItem($fields[$i],
$UserName);
        $returnValue .= $message_dbmessage[0];
        $checked++;

        if ($message_dbmessage[1] != 0)
        {
            $oDBH->commit();
        }
        else
        {
            $oDBH->rollback();
        }
    }
}

if ($checked>0)
{
    ##show the page that tells user if space was cancelled or not
    &ShowCanceled($returnValue,$oTemplate);
}
else
{
    ##user hasn't checked anything, give em error page
    my $sError = $oErr->ReturnMessageGivenCode(1301);
    XDErrorToBrowser("", $sError, undef, $oToken);
}
}

```

```

#####
## ShowCanceled: tell user space was cancelled
#####

```

```

sub ShowCanceled
{
    my $sItemsCanceled = shift;
    my $oTemplate       = shift;

    ## Load the required template HTML files.
    $oTemplate->load('removespace_ok.html');
    $oTemplate->tags
    (
        {
            'items' => $sItemsCanceled
        }
    );
    print "Content-type: text/html\n\n";
    print $oTemplate->get();
}

```

```

#####
## ShowSpace: shows the user the initial page with their current space
## allocation
#####

```

```

sub ShowSpace
{

```



```

my $sUserName = shift;
my $oTemplate = shift;
my $oToken = shift;
my $oDBH = shift;
my $oErr = shift;

my $sMessage = $oErr->ReturnMessageGivenCode(1302);
$sMessage = &GetItems($sUserName,$oToken,$oDBH,$oErr);

## Load the required template HTML files.
$oTemplate->load('removespace_request.shtml');
$oTemplate->tags
(
    (
        'items' => $sMessage
    )
);

print "Content-type: text/html\n\n";
print $oTemplate->get();
}

sub ShowIntroPage
{
    my $oTemplate = shift;
    my $sUserName = shift;
    my $oToken = shift;
    my $oCGI = shift;

    my $oAction = new XDrive::Client::Actions
    (
        $oToken,
        $oCGI
    );
    my $quotaUsed = $oAction->QuotaUsed();
    $quotaUsed = sprintf("%2.2f",$quotaUsed/1024);
    my $quotaLimit = $oAction->QuotaLimit();
    $quotaLimit = sprintf("%2.2f",$quotaLimit/1024);

    $oTemplate->load('removespace_intro.shtml');
    $oTemplate->tags
    (
        (
            'quotaUsed' => $quotaUsed,
            'quotaLimit' => $quotaLimit
        )
    );
    $oTemplate->clear();

    print "Content-type: text/html\n\n";
    print $oTemplate->get();
}

sub GetItems
{
    my $sUserName = shift;
    my $oToken = shift;
    my $oDBH = shift;
    my $oErr = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

    ##now load all items in the user_purchase database that are
    ##owned by this user
    my $userSeq = $oDiskAccount->fetchColumn('USER_SEQ');

```

```

##passing a 0 as the last parameter returns all non-canceled items
my $oSearch = XDrive::DatabaseO::Search->new(undef);
my $array = $oSearch->XDUserPurchases($userSeq,0);

##see if the array returned any items

if ($array->[0][0] eq '')
{
    my $sError = $oErr->ReturnMessageGivenCode(1302);
    XDErrorToBrowser('removespace_noitems.shtml', $sError, 1,
$oToken);
}

my $i;
my $items = '';

for $i(0..$#{ $array })
{
    ##storing the complete string returned by Epoch
    ##must take only stuff after the | to cancel transaction
    ##and chop off last character which seems to be a line return
    ##may have to alter this if we see problems
    chop($array->[$i][4]);
    my @aCodes=split(/\|/, $array->[$i][4]);
    my $itemName = 'tc_' . $aCodes[1];
    $itemName=~s/~///;

    ##Get the name associated with this item
    my $oDeal = XDrive::DatabaseO::Table::Deal->new(undef,$oDBH);
    $oDeal->loadWhere('SEQ',$array->[$i][2]);
    my $itemSeq = $oDeal->fetchColumn('ITEM_SEQ');
    my $oItem = XDrive::DatabaseO::Table::Item->new(undef,$oDBH);
    $oItem->loadWhere('SEQ', $itemSeq);
    my $description = $oItem->fetchColumn('DESCRIPTION');

    $items .= '<input type="checkbox" name="' . $itemName . '">' .
$description . '<BR>';
}

if ($items eq '')
{
    my $sError = $oErr->ReturnMessageGivenCode(1302);
    XDErrorToBrowser('removespace_noitems.shtml', $sError, 1,
$oToken);
}

return $items;
}

```

###selected_delete.cgi

```
#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> for renaming files from the
# web.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::Client::Security;

use XDrive::Error;
use XDrive::DatabaseO;

exit &main;

sub main
{
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    #####
    ## Attempt to autenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If an error occured or the user could not be validated then
    ## redirect to the error CGI and exit
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    #####
    ## Otherwise we know that we have a valid session
    #####

    my $oAction = new XDrive::Client::Actions
    (
        $oToken,
        $oCGI
    );

    $oAction->FileCheck($oAction->ItemCurrent());
    $oAction->ItemDelete($oAction->ItemCurrent());
    xd_web_buttonindex($oCGI);
    $oAction->DisconnectDB();

    return 0;
}
```

###selected_rename.cgi

```
#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> for renaming files from the
# web.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp 'fatalToBrowser';
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::Client::Security;

use XDrive::Library;
use XDrive::DatabaseO;
use XDrive::Error;

## Clean up the path
$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)); # Make %ENV safer

exit &main;

sub main {
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    ####
    ## Attempt to authenticate the user
    ####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    ####
    ## If the authentication fails or there is an error during the
    ## authentication phase then redirect to the error CGI
    ####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    ####
    ## Otherwise we have a valid session
    ####

    my $oAction = new XDrive::Client::Actions
    (
        $oToken,
        $oCGI
    );

    my $sItemOld = $oAction->ItemCurrent();

    ## Get the relative path to the item to be renamed from the
    ## old item name itself.

```

```
my ($sFolder) = $sItemOld =~ /(.\+\/){^\+\/}+;/

## Set the new item to be in that folder.
my $sItemNew = $sFolder.$oAction->ItemNew().$oAction-
>ItemExtension();

$oAction->FileCheck($sItemOld);
$oAction->ItemRename($sItemOld,$sItemNew);

xd_web_buttonindex($oCGI);
$oAction->DisconnectDB();
)
```

###settings_save.cgi

```
#!/usr/bin/perl

use strict;
use vars qw(@ISA);
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use Data::Dumper;
use XDrive::Library;
use XDrive::CGI;
use XDrive::Client::Quota;
use XDrive::Client::Security;
use XDrive::CGI::Cookie;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::Language;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::Template;

@ISA = qw(XDrive::CGI);

exit &main;

sub main {
    my $oCGI = CGI->new();
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    #####
    ## Attempt to autenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If the autentication fails or there is an error during the
    ## autentication phase then redirect to the error CGI
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    #####
    ## Otherwise we have a valid session
    #####

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $sUser = $oToken->data('user');
    my $nUser = UserIdGet($sUser);

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings->new(undef,
undef);

    ## Initialize global variables
    my $g_bFileExtEdit = $oCGI->param('bFileExtEdit') eq 'on' ? 1 : 0;

```

```

my $g_bExtraHelp    = $oCGI->param('bExtraHelp') eq 'on' ? 1 : 0;
my $g_bMarketing    = $oCGI->param('bMarketing') eq 'on' ? 1 : 0;
my $g_bNewsletter   = $oCGI->param('bNewsletter') eq 'on' ? 1 : 0;
my $g_bLanguage     = $oCGI->param('bLanguage');
my $sCurrentLanguage;

my $languageCode;

if (defined $g_bLanguage)
{
    my $oLanguage = XDrive::DatabaseO::Table::Language->new
        (undef, $oUserSettings->fetchDBO());
    $oLanguage->loadWhere("CODE", $g_bLanguage);
    $languageCode = $oLanguage->fetchColumn("SEQ");
}

## We are doing this in a backwards way -- first we will try and load
the ## current users profile.  If that works then we change it and update
it ## by calling save.  If that does not work then we just call save.

$oUserSettings->loadWhere("USER_SEQ", $nUser);
$oUserSettings->setColumn("FILE_EXT_EDITABLE", $g_bFileExtEdit);
$oUserSettings->setColumn("EXTRA_HELP", $g_bExtraHelp);
$oUserSettings->setColumn("OPT_MARKETING", $g_bMarketing);
$oUserSettings->setColumn("OPT_NEWSLETTER", $g_bNewsletter);

## The language element is an OPTIONAL setting in the "My Profile"
area. ## If it is passed then set it, otherwise leave the current value.
if (defined $g_bLanguage)
{
    $sCurrentLanguage = $g_bLanguage;
    $oUserSettings->setColumn("LANGUAGE", $languageCode);
}
else
{
    $sCurrentLanguage = "english";
}

my $status = $oUserSettings->update();

if ($status < 0)
{
    $oUserSettings->rollback();
    my $sMessage = $oErr->ReturnMessageGivenCode(1330);
    XDErrorToBrowser(undef, $sMessage, undef, $oToken)
}
else
{
    $oUserSettings->commit();

    if (defined $g_bLanguage)
    {
        ##set the cookie for language
        $oCookie->setElement
        (
            'language' => $g_bLanguage
        );
        print "Set-Cookie: ", $oCookie->asString();
    }
}

```

```
## Redirect the browser to the succesfull save page.

### Edited by Justin so that we get the partner_code out
### of cookie instead of the token table.
# print xd_web_redirect
#   (
#     "/account/profile/$sCurrentLanguage/saved.html",
#     $oToken->data('partner_code')
#   );

print xd_web_redirect
(
  "/account/profile/$sCurrentLanguage/saved.html",
  $oCookie->getElement('partner')
);

$oUserSettings->finish();
$oUserSettings->disconnect();
$oDBO->disconnect();
}
```


###share_a_file.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Client::Quota;
use Math::TrulyRandom;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Search;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::UserData;
use XDrive::Utils::RandomString;
use XDrive::CGI;

use Mail::Sendmail;
use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Template;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Library;
use XDrive::CGI::Cookie;

use strict;

&main();

sub main {
    my $cgi = CGI->new();
    my $oErr = new XDrive::Error;
    my $xdDBH = XDrive::DatabaseO->new();

    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $cgi);

    #####
    ## Attempt to authenticate the user
    #####

    my $oToken = xd_security_check($xdDBH,$cgi,$oErr);

    #####
    ## If the authentication fails or there is an error during the
    ## authentication phase then redirect to the error CGI
    #####

    if ($oErr->Occurud) {
        xd_fatal_error($cgi,$oErr);
        $xdDBH->disconnect();
        exit;
    }

    #####
    ## Otherwise we have a valid session
    #####

    ### Edited by Justin so that the partner_code is looked for
    ### in the cookie instead of the token table.
    # my $sPartner = $oToken->data('partner_code');
    my $sPartner = $oCookie->getElement('partner');
    my $nUser_ID = UserIdGet($oToken->data('user'));

```

```

## Grab the user info from the Database
my $oUserInfo = XDrive::DatabaseO::Table::UserData->new({}, $xdDBH);

my $sFileName = $cgi->param("sFileName");
my $bHelp = $cgi->param("help");

my $sFriendsEmail = &get_friends_emails($cgi);
my $sEmailSubject = $cgi->param('sEmailSubject');
my $sEmailMessage = $cgi->param("sEmailMessage");
my $sFileDescription = $cgi->param("sFileDescription");

my ($sRandomKey, $sFilePath);

## Load user info where the SEQ = $nUser_ID
$oUserInfo->loadWhere("SEQ", $nUser_ID);

my $sUser_name = $oUserInfo->fetchColumn("NAME_FIRST") . " " .
$oUserInfo->fetchColumn("NAME_LAST");
my $sUser_email = $oUserInfo->fetchColumn("EMAIL_ADDRESS");

if ($sFriendsEmail)
{
    $sFilePath="/";
    $sFileName =~ m%(.*)/(.*)%;

    #inserted this code to catch documents that are not in a folder

    my $tempFilePath = "/" . $1;
    my $tempFileName = $2;

    if ($tempFileName ne "")
    {
        $sFileName=$tempFileName;
        $sFilePath=$tempFilePath;
    }

    &verify_database_values($nUser_ID, $sFileName, $sFilePath,
        $sFilePath, $sFileName,
        $sFileDescription,$oToken,$oErr);

    ## Insert the info into the disk_item_share table, and get the
    random key
    $sRandomKey = &insert_file_into_database($nUser_ID, $sFileName,
        $sFilePath, $sFileDescription, $xdDBH,$oToken,$oErr);

    &send_mail($sFriendsEmail, $sEmailSubject, $sEmailMessage, $sFileDescription,
        $sUser_name, $sUser_email, $nUser_ID,
        $sRandomKey,$sPartner,$oToken,$oErr,$cgi);

    &display_thank_you($sPartner);
}
else {
    $oUserInfo->finish();
    $xdDBH->disconnect();
    &display_form($sFileName, $bHelp, $sPartner);
}

$oUserInfo->finish();
$oUserInfo->disconnect();
}

```

```

sub send_mail {
    my ($sFriendsEmail, $sEmailSubject, $sEmailMessage, $sFileDescription,
        $sUser_name, $sUser_email, $nUser_ID, $sRandomKey,
        $sPartner, $oToken, $oErr, $oCGI) = @_;

    ##get language from the cookie. If not english, append language code
    to url
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);
    my $language = $oCookie->getElement('language');
    if ($language ne 'english')
    {
        if ($language eq 'spanish')
        {
            $sRandomKey .= "--SP";
        }
    }

    $sEmailMessage = &get_message($sEmailMessage,
        $sRandomKey, $sPartner, $nUser_ID);

    my %toXdrive =
    (
        To      => "$sFriendsEmail",
        From     => "$sUser_name.<$sUser_email>",
        Message => $sEmailMessage,
        Subject => "$sEmailSubject",
    );

    unless (sendmail %toXdrive)
    {
        warn "## Mail error ".$Mail::Sendmail::error;
        if ($Mail::Sendmail::error =~ /451/)
        {
            my $sMessage = $oErr->ReturnMessageGivenCode(1310);
            XDErrorToBrowser("", $sMessage, undef, $oToken);
        }
        else
        {
            my $sMessage = $oErr->ReturnMessageGivenCode(1311);
            XDErrorToBrowser("", $sMessage, undef, $oToken);
        }
        exit(1);
    }
}

sub get_message {
    my ($sEmailMessage, $sRandomKey, $sPartner, $n_UserID) = @_;

    my $oMessage = new XDrive::Template;
    $oMessage->partner($sPartner);
    $oMessage->load('share_a_file__message.thtml');

    $oMessage->tags
    (
        {
            'Message' => $sEmailMessage,
            'RandomKey' => $sRandomKey,
            'nUser_ID' => $n_UserID,
            'sender' => $ENV{'HTTP_HOST'},
        });

    return $oMessage->get;
}

```

```

}

sub display_form {
    my ($sFileName, $bHelp, $sPartner) = @_;
    my $oForm = new XDrive::Template;
    $oForm->partner($sPartner);
    $oForm->load('share_a_file.thtml');

    my $sHelp='';

    if ($bHelp eq 'true')
    {
        my $oHelp = new XDrive::Template;
        $oHelp->partner($sPartner);
        $oHelp->load('share_a_file_help.thtml');
        $sHelp = $oHelp->get;
    }

    $oForm->tags
    (
        {
            'sFileName' => $sFileName,
            'helptext' => $sHelp
        }
    );

    print header, $oForm->get;
    exit(0);
}

sub display_thank_you {
    my $sPartner = shift;
    my $oForm = new XDrive::Template;
    $oForm->partner($sPartner);
    $oForm->load('share_a_file__t_y.thtml');
    print header, $oForm->get;
    exit(0);
}

sub verify_database_values {
    my ($nUser_ID, $sFileName, $sFilePath, $sFilePath, $sFileName,
    $sDescription, $oToken, $oErr) = @_;

    if (length($sDescription) > 255) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1320);
        XLErrorToBrowser("", $sMessage, undef, $oToken);
    }

    if (length($sFilePath) > 255) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1321);
        XLErrorToBrowser("", $sMessage, undef, $oToken);
    }

    if (length($sFileName) > 255) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1322);
        XLErrorToBrowser("", $sMessage, undef, $oToken);
    }
}

sub insert_file_into_database {
    my ($nUser_ID, $sFileName, $sFilePath, $sFileDescription,
    $xdDBH, $oToken, $oErr) = @_;

```

```

my @characters = ('a'..'z','A'..'Z','0'..'9');

##seed random number generator
srand(truly_random_value());
my $gmTime = time;
##grab length of time
my $randLen = 32 - length($gmTime);
my $sRandomKey = XDRandomString($randLen,\@characters);
##now we have a Random key
$sRandomKey = $gmTime . $sRandomKey;
## at this point we have a random number
## of length gmTime with the current gmt time appended to it

my $stransaction = XDrive::DatabaseO::Transaction->new($xdDBH);
my $status = $stransaction->insertDiskItemShare($nUser_ID, $sRandomKey,
$sFilePath, $sFileName, $sFileDescription);

if ($status < 0)
{
    $stransaction->rollback();
    my $sMessage = $oErr->ReturnMessageGivenCode(1323);
    XDErrorToBrowser("", $sMessage, undef, $oToken);
    exit(1);
}
else
{
    $stransaction->commit();
}

return $sRandomKey;
}

sub get_friends_emails {
my $cgi = shift;
my ($email_list, @email_array);

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail0'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail1'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail2'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail3'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail4'));
}
}

```

```
$email_list = join(",", @email_array);  
return $email_list;  
}
```

###signup_account.cgi

```

#!/usr/bin/perl
## -d:DProf
## -d:SmallProf
## Written by Martin Hald <mhald@uci.edu> on Wed Apr 7 1999. This program
## adds new users to the database.
## Modified by Justin White for cookie referee and promo stuff and to make
## mod_perl friendly and to work with changes to the Security module and
## to get rid of the XDrive::CGI module and to create a CGI object.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::Client::Registration;
use XDrive::Error;
use XDrive::Client::Security;
use XDrive::Template;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::UserQuota;
use XDrive::DatabaseO::Table::Promo;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::Reseller;
use XDrive::Template;
use XDrive::DatabaseO::Search;
use XDrive::CGI::Cookie;
use XDrive::Library;
use Mail::Sendmail;
use CGI qw(param redirect header cookie);

BEGIN
{
    push(@INC, "/export/home/www/thirdparty/mint2/perl");
}

use Mint2;

&main;

exit;

sub main {
    my $oCGI = new CGI;
    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $oSTDCookie = XDrive::CGI::Cookie->new('xd_std_info', $oCGI);

    my $file_found;

    ### Use the new XDrive::CGI::Cookie now.
    my $promo_uri = $oCookie->getElement('promo');
    my $ref_seq_cookie = $oCookie->getElement('referee');
    my $referred_from = $oCookie->getElement('referred_from');
    my $claim_ticket = $oCookie->getElement('ct');

    my $ref_seq_param = $oCGI->param('referee');
    my $password = $oCGI->param('password');
    my $password_confirm = $oCGI->param('password_confirm');
    my $birth_year = $oCGI->param('birth_year');

```

```

my $username      = $oCGI->param('username');
my $name_first    = $oCGI->param('name_first');
my $name_last     = $oCGI->param('name_last');
my $email_address = $oCGI->param('email_address');
my $country_seq   = $oCGI->param('country');
my $gender_seq    = $oCGI->param('gender');
my $postal_code   = $oCGI->param('zip2');
my $occupation_seq = $oCGI->param('occupation');
my $referee       = $oCGI->param('referee');
my $marketing     = $oCGI->param('marketing');
my $newsletter    = $oCGI->param('newsletter');
my $media_type_seq = $oCGI->param('media_type');

## PARAMS TO GATHER IF THIS IS CALLED FROM SKIP
## THE DOWNLOAD
my $sSTDPartner = $oSTDCookie->getElement('STDPARTNER');
my $sLanguage   = $oSTDCookie->getElement('LANG');
my $sFileURL    = $oSTDCookie->getElement('FILEURL');
my $sFileName   = $oSTDCookie->getElement('FILENAME');
my $sAltURL     = $oSTDCookie->getElement('ALTURL');
my $sCatId      = $oSTDCookie->getElement('CATID');
my $sGid        = $oSTDCookie->getElement('GID');
my $sSid        = $oSTDCookie->getElement('SID');

## check if database is up
my $oDBO;
my $oSearch;
if (XDDDBConnectionCheck() && XDNFSCheck())
{
    ## connection good proceed normally
    $oDBO = new XDrive::DatabaseO(undef);
    $oSearch = XDrive::DatabaseO::Search->new($oDBO);
}
else
{
    ## connection bad write data to a temp file and load
    ## upgrading page telling them that they will be
    ## informed once X:drive is up
    $oDBO = undef;
    $oSearch = undef;
    my $tempVar;

    my $tempEmail = $oCGI->param('friends_email1');
    my $numFriends = $oCGI->param('numFriends');

    my $addrArray = $tempEmail;
    my $nameArray = $oCGI->param('friends_name1');

    ## generate list for the javascript array
    for (my $i = 2; $i <= $numFriends; $i++)
    {
        $tempVar = $oCGI->param('friends_email' . $i);

        if ($tempVar)
        {
            $addrArray .= "~" . $tempVar;
            $nameArray .= "~" . $oCGI->param('friends_name' . $i);
        }
    }
}

```



```

    reg_while_down (
        $promo_uri,
        $ref_seq_cookie,
        $referred_from,
        $claim_ticket,
        $ref_seq_param,
        $password,
        $birth_year,
        $username,
        $name_first,
        $name_last,
        $email_address,
        $country_seq,
        $gender_seq,
        $postal_code,
        $occupation_seq,
        $referee,
        $marketing,
        $newsletter,
        $media_type_seq,
        $nameArray,
        $addrArray
    );

    ## leave and show upgrading page test me
    print redirect("/upgrading_signup_success.html");
    exit;
}

##my $oDBO      = new XDrive::DatabaseO(undef);
##my $oSearch = XDrive::DatabaseO::Search->new($oDBO);

#####
### If media_type_seq equals 'notset', then set it to NULL.
#####
$media_type_seq = '' if $media_type_seq eq 'notset';

my $partner_code = 'xdrv';
my $partner_seq  = 1;

my $promo_seq;

#####
### Check to see how the referee sequence, if any, was passed in.
### If it was passed in via cookie, then use that. Else, assume
### that it is a form parameter.
#####
my $ref_seq = $ref_seq_cookie ? $ref_seq_cookie : $ref_seq_param;

#####
### If we were passed a promo uri, then let's get the promo seq
### from promo table using Promo.pm to pass to xd_client_register.
#####
if ($promo_uri) {
    my $oPromoInfo = XDrive::DatabaseO::Table::Promo-
>new(undef,$oDBO);
    $oPromoInfo->loadWhere('URI', $promo_uri);
    $promo_seq = $oPromoInfo->fetchColumn('SEQ');

    $oPromoInfo->finish();
}

#####

```

```

    ### Load the required template HTML files. The content that we load
depends
    ### on if the new registration went through or if we need to have them
re-fill
    ### the form.
    #####
    my $oContent      = new XDrive::Template( {'partner_code' => 'xdrv'} );
    my $oLayout       = new XDrive::Template( {'partner_code' => 'xdrv'} );
    my $oNavigation   = new XDrive::Template( {'partner_code' => 'xdrv'} );

    my $oErr = new XDrive::Error;

    $oContent->load('front_signup.thtml');
    $oNavigation->load('front_nav.thtml');
    $oLayout->load('layout.thtml');

    #####
    ### Perform data validation
    #####
    if ($password ne $password_confirm) {
        $oErr->AddErrorByErrorCode(709);
    }

    #####
    ### Attempt to register the user if no errors have been logged
    #####
    if (! $oErr->Occurud ) {
        xd_client_register( {'birth_year'      => $birth_year,
                             'partner_seq'     => $partner_seq,
                             'username'        => $username,
                             'password'        => XDEncrypt($password),
                             'name_first'      => $name_first,
                             'name_last'       => $name_last,
                             'email_address'    => $email_address,
                             'country_seq'     => $country_seq,
                             'gender'          => $gender_seq,
                             'postal_code'     => $postal_code,
                             'occupation_seq'   => $occupation_seq,
                             'referee'         => $ref_seq,
                             'marketing'       => $marketing,
                             'newsletter'      => $newsletter,
                             'partner_code'     => $partner_code,
                             'promo_seq'       => $promo_seq,
                             'media_type_seq'   => $media_type_seq},
        $oCGI, $oErr, $oDBO );
    }

    if ($oErr->MaxIndex() < 0) {
        ## No errors occurred, the user has already been added to the
        ## database through the xd_client_register subroutine so now
        ## send the user an email and then
        ## log the user and go to the user's homepage.

        client_email_send($username,
                          $name_first,
                          $name_last,
                          $email_address,
                          'X\drive Team <team@xdrive.com>',
                          'Welcome to X:drive! - Important Account
Information',
                          $partner_code,
                          $promo_seq);
    }

```

```

#####
### If we have a claim ticket, then remove that ticket
### from the batch_user_data table because the user has
### been added and we don't need that data anymore.
#####
if ($claim_ticket) {
    my $oTransaction = XDrive::DatabaseO::Transaction-
>new($oDBO);

    my $rv = $oTransaction->removeClaimTicket($claim_ticket);

    if ($rv == 1) {
        $oTransaction->commit();
    }
    else {
        $oTransaction->rollback();
    }
}

##if we have a referee seq, give the referee additional space
if ($ref_seq >= 1) {

    ## johngaa add to exclude college club and quepasa users
    out
    my $oUserData = XDrive::DatabaseO::Table::UserData-
>new(undef,$oDBO);
    $oUserData->loadWhere("SEQ", $ref_seq);
    my $reseller_seq = $oUserData-
>fetchColumn("RESELLER_SEQ");
    if (!(isResellerSeqCC_QUPA($oDBO,$reseller_seq)))
    {

        ## end of johngaa

        my $oUserQuota = XDrive::DatabaseO::Table::UserQuota-
>new(undef, $oDBO);
        $oUserQuota->loadWhere("USER_SEQ", $ref_seq);
        my $additional_quota = $oUserQuota-
>incrementQuota($ref_seq, 5120);

        if ($additional_quota > 0) {
            &send_email_referee($ref_seq,$oDBO,$oCookie,$additional_quota,$referred_from)
            ;

        }

        $oUserQuota->finish();
    }

}

##if the user is from Cybergold, process through Cybergold
if ($promo_uri =~ /cybergold/) {
    my ($code, %res) =
&contact_cybergold($oCGI,$username,$email_address);
}

##if user is coming from the befree promo
##write to file that they've signed up
if ($promo_uri =~ /befree/) {
    &write_befree_log($oCGI);
}

```

```

    }

    if ($sFileURL eq '') {

        client_login($username, $oCGI);

    } else {

        std_login($username,
            $oCGI,
            $sSTDPartner,
            $sLanguage,
            $sFileURL,
            $sFileName,
            $sAltURL,
            $sCatId,
            $sGid,
            $sSid);

    }

    $oSearch->disconnect();

    exit;
}
else {
    ## Reload the signup form, show the errors and pre-fill all
    ## the form elements except the password.

    ##if we are overriding standard registration form
    ##load it here

    if ($promo_uri)
    {
        $file_found = $oContent->load($promo_uri .
'_registration.shtml');
        if (!$file_found)
        {
            $file_found = $oContent-
>load('promo_registration.shtml');
        }
    }

    if ((!$promo_uri) || (!$file_found))
    {
        $oLayout->load("layout.shtml");
        $oNavigation->load("front_nav.shtml");
        $oContent->load("front_signup.shtml");
    }

    my ($select_marketing, $select_newsletter);

    my $checked = "CHECKED";

    if ($marketing eq 'on') {
        $select_marketing = $checked;
    }

    if ($newsletter eq 'on') {
        $select_newsletter = $checked;
    }

    ## IMPORTANT ##
    ## make sure to put all non text fields at the top of

```

```

## the tags function or it will gag

## Search and replace the following tags
$soContent->tags( {'country'      =>
xd_form_countries($country_seq, $soSearch),
                  'occupation'   =>
xd_form_occupation($occupation_seq, $soSearch),
                  'media_type'   =>
xd_form_media_type($media_type_seq, $soSearch),
                  'gender'       =>
xd_form_gender($gender_seq, $soSearch),
                  'select_marketing' => $select_marketing,
                  'select_newsletter' => $select_newsletter,
                  'errors'        => format_errors($soErr),
                  'username'      => $username,
                  'name_first'    => $name_first,
                  'name_last'     => $name_last,,
                  'email_address' => $email_address,
                  'birth_year'    => $birth_year,
                  'postal_code'   => $postal_code) );

##
## Added to have tell a friend support in registration
##

my (@addrArray, @nameArray, $tempIndex, $tempName,
$tempEmail, $tempNum);

## tell a friend data will be coming in to signup_form
## seperated by commas

@addrArray = split /,/, $soCGI->param('friends_email_array');
@nameArray = split /,/, $soCGI->param('friends_name_array');

$tempNum = $soCGI->param('numFriends');

for (my $tempIndex=1; $tempIndex <= $tempNum; $tempIndex++) {
    $tempName = 'friends_name' . $tempIndex;
    $tempEmail = 'friends_email' . $tempIndex;

    $soContent->tags( {$tempName => $soCGI->param($tempName),
                      $tempEmail => $soCGI->param($tempEmail)}
);

}

## Clear the content of any unused tags.
$soContent->clear;
}

##if we are loading a non-standard registration, it's only one page
if (($promo_uri) && ($file_found))
{
    print $soCGI->header(), $soContent->get;
}
else
{
    ## Print out the HTML and exit
    $soLayout->tags( {'header_graphic' => 'header_registration.gif',
                    'title'           => 'Register Now!',
                    'content'         => $soContent->get,
                    'navigation'      => $soNavigation->get} );
}

```

```

        print $oCGI->header(), $oLayout->get;
    }

    $oSearch->disconnect();

    return 0;
}

```

```

sub isResellerSeqCC_QUPA
{
    my $oDBO = shift;
    my $reseller_seq = shift;
    my $dbh = $oDBO->fetchDBH();

    my $sql_stmt = "SELECT code FROM reseller WHERE seq=?";
    my $cmd;
    my @data;

    $cmd = $dbh->prepare($sql_stmt);
    $cmd->execute(($reseller_seq));
    @data = $cmd->fetchrow_array;
    if ($data[0] eq 'cc' || $data[0] eq 'qupa')
    {
        return 1;
        ##print "should return a true\n"
    }
    return 0;
}

```

```

#####
## reg_while_down:  Grabs all data that is needed to register a user
## routine will add the data to a file in the tmp directory of the name
## reg_while_down.datetime
#####
sub reg_while_down

```

```

{
    my ($promo_uri,$ref_seq_cookie,$referred_from,$claim_ticket,
        $ref_seq_param,$password,$birth_year,$username,$name_first,
        $name_last,$email_address,$country_seq,$gender_seq,$postal_code,
        $occupation_seq,$referee,$marketing,$newsletter,$media_type_seq,
        $tell_a_friend_name,$tell_a_friend_addr) = @_;

    my $filename = XDGetRegDatFile();
    open OUTFILE, ">>$filename";

    print OUTFILE "$promo_uri,$ref_seq_cookie,$referred_from,";
    print OUTFILE "$claim_ticket,$ref_seq_param,$password,";
    print OUTFILE "$birth_year,$username,$name_first,";
    print OUTFILE "$name_last,$email_address,$country_seq,";
    print OUTFILE "$gender_seq,$postal_code,$occupation_seq,";
    print OUTFILE "$referee,$marketing,$newsletter,$media_type_seq,";
    print OUTFILE "$tell_a_friend_name,$tell_a_friend_addr\n";
    close OUTFILE;
}

```

```

#####
## format_errors:  Accept an error object and return an ordered list of
## errors in HTML format.
#####

```

```

sub format_errors {
    my $oErr = shift; ## (I) errors

    my $txt;      ## formatted HTML
    my $bPassword; ## has a password error been found?

    $txt .= "<ol>\n";

    my $nNumErrors = $oErr->MaxIndex();

    for (my $i = 0; $i <= $nNumErrors; $i++) {
        my $error = $oErr->Message();

        if ($error =~ /assword/) {
            $bPassword = 1;
        }

        $txt .= "<li><font color=RED>$error</font>\n";
    }

    if (! $bPassword) {
        $txt .= "<li><font color=RED>Please re-enter your
password</font>\n";
    }

    $txt .= "</ol>\n";

    return $txt;
}

#####
## client_login: Create the needed token to identify the client and redirect
## them to thier new homepage.
#####

sub client_login ($$) {
    ## No errors occurred, add the user to the parter/user->real
    ## user mapping and return a success code.

    my $username = shift;
    my $oCGI      = shift;

    my $oDBO      = new XDrive::DatabaseO(undef);
    my $oCookie   = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    #####
    ### Check the x_session_info cookie for promo or referee and
    ### if they exist, delete those hash elements and reset the cookie.
    #####

    my $promo_cookie = $oCookie->getElement('promo');
    my $ref_cookie   = $oCookie->getElement('referee');

    if ($ref_cookie || $promo_cookie) {
        $oCookie->deleteElement('referee') if $ref_cookie;
        $oCookie->deleteElement('promo')   if $promo_cookie;

        print "Set-Cookie: ", $oCookie->asString();
    }

    my $oError = new XDrive::Error;
    my $oToken = xd_login($oCGI, $username, $oError, $oDBO);

```

```

## we need to do all of this to get the reseller code to show the
correct page
my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new(undef,
$oDBO);

    $oDiskAccount->loadWhere("USERNAME", $username);

    my $oUser = XDrive::DatabaseO::Table::UserData->new(undef,
$oDiskAccount->fetchDBO);

    $oUser->loadWherePK($oDiskAccount->fetchColumn("USER_SEQ"));

    my $oReseller = XDrive::DatabaseO::Table::Reseller->new(undef,
$oDiskAccount->fetchDBO);

    $oReseller->loadWherePK($oUser->fetchColumn("RESELLER_SEQ"));

my $oTemplate = new XDrive::Template;

    $oTemplate->partner($oReseller->fetchColumn("CODE"));

    ## originally this is where the signup_form.cgi goes
##$oTemplate->load('splash.shtml');
$oTemplate->load('tell_a_friend_frame.shtml');

    ##my $addrArray = $oCGI->param('friends_email_array');
    ##my $nameArray = $oCGI->param('friends_name_array');
    ##my $numFriends = $oCGI->param('numFriends');

    ## generate list for the javascript array
    ##my @addrList = split /,/, $addrArray;
    ##my @nameList = split /,/, $nameArray;

    ##$addrArray = "";
    ##$nameArray = "";

    ##my $count = @addrList - 1;

    ##for (my $i = 0;$i < $count;$i++) {
        ##$addrArray .= "\"\" . $addrList[$i] . "\",";
        ##$nameArray .= "\"\" . $nameList[$i] . "\",";
    ##}
    ## this will add the quote without the comma

    ##$addrArray .= "\"\" . $addrList[$count] . "\"";
    ##$nameArray .= "\"\" . $nameList[$count] . "\"";
    ## gets the array started
my $tempVar;

my $tempEmail = $oCGI->param('friends_email1');
my $numFriends = $oCGI->param('numFriends');

my $addrArray = "\"\" . $tempEmail . "\"";
my $nameArray = "\"\" . $oCGI->param('friends_name1') . "\"";

## generate list for the javascript array
for (my $i = 2;$i <= $numFriends;$i++)
{
    $tempVar = $oCGI->param('friends_email' . $i);

    if ($tempVar)
    {

```



```

    $addrArray .= ", \" . $tempVar . "\"";
    $nameArray .= ", \" . $oCGI->param('friends_name' . $i) . "\"";
}
}

$oTemplate->tags( {'numFriends' => $numFriends,
                  'friends_name_array' => $nameArray,
                  'friends_email_array' => $addrArray} );

print $oCGI->header();

print $oTemplate->get();

$oDiskAccount->finish();
$oUser->finish();
$oReseller->finish();
$oDiskAccount->disconnect();
}

#####
## Login in user who is coming from a Skip The Download
## Registration
#####

sub std_login () {
    my $username = shift;
    my $oCGI = shift;
    my $sSTDPartner = shift;
    my $sLanguage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sCatId = shift;
    my $sGid = shift;
    my $sSid = shift;
    my $oDBO = new XDrive::DatabaseO(undef);

    my $oError = new XDrive::Error;
    my $oToken = xd_login($oCGI, $username, $oError, $oDBO);
    xd_set_session_cookie($oCGI, $sSTDPartner, $sLanguage);

    my $oTemplate = new XDrive::Template
    (
        'partner_code' => $sSTDPartner,
        'language' => $sLanguage,
        'file' => 'skip_the_download_from_reg.shtml',
        'tags' =>
        {
            'FILE_URL' => $sFileURL,
            'FILE_NAME' => $sFileName,
            'ALTRUL' => $sAltURL,
            'LANG' => $sLanguage,
            'STDPARTNER' => $sSTDPartner,
            'CATID' => $sCatId,
            'GID' => $sGid,
            'SID' => $sSid,
        }
    );
};

```

```

    $oTemplate->clear();
    print "Content-type: text/html\n\n";
    print $oTemplate->get();
    $oDBO->disconnect();
}

sub contact_cybergold {
    my $oCGI = shift;
    my $msgid = shift;
    my $email = shift;

    my %args = (
        'mint_home'    => $ENV{'MINT_HOME'},
        'msg_mode'     => 'background_mode',

        'usr_email'    => $email,
        'msg_id'       => $msgid,

        'pay_type'     => 'reward',
        'pay_value'    => '1.00',
        'pay_readme'   => 'Thanks for registering with X:drive.',

        'co_name'      => 'X Drive',
        'co_key'       => 'registration',
        'co_account'   => '100500900000396',
        'mint_secret'  => '184FEB9DB81944502A1C91B2879484B6',

        'mint_url_pay' => 'http://www1.cybergold.com/payserver?pay_server',
        'msg_version'  => '2.2'
    );

    my($code, %res) = mint_invoke(\%args);

    ##this is temp code to print out stuff for cybergold
    ##my @keys = keys %res;
    ##my @values = values %res;
    ##while (@keys)
    ##{
    ##    die pop(@keys), '=', pop(@values), "\n";
    ##}

    return $code;
}

sub write_befree_log {
    my $oCGI = shift;

    my $source_id = $oCGI->cookie('sourceid');

    ##get the time
    ##needed to figure out name of file to write to
    my ($nSec, $nMin, $nHour, $nDay, $nMonth, $nYear, $sDay) =
        (localtime(time))[0,1,2,3,4,5,6];

    if ($nYear > 99) {
        $nYear = substr($nYear,1,2);
    }

    ## Numeric month is 0-11, so add one
    $nMonth++;

    ## Handle Y2K issue

```

```

        if ( $nYear >= 80 ) {
            $nYear += 1900;
        }
        else {
            $nYear += 2000;
        }

        my $dToday = sprintf("%s%02d%02d", $nYear, $nMonth, $nDay);
        my $dTodayFull = sprintf("%02d%02d%02d:%02d:%02d", $nMonth, $nDay, $nYear, $nHour, $nMin, $nSec);

        my $text =
"14524098\tS\t$dTodayFull\t$source_id\tl\tl\tl\t0.00\tUSD\tregistration\n";

        warn "#BF", $text, "\n";
        ##open(FILE, ">>xdrive_orders_$dToday.txt");
        ##print FILE $text;
        ##close(FILE);
    }

sub send_email_referee {
    my $user_seq = shift;
    my $oDBO = shift;
    my $oCookie = shift;
    my $additional_quota = shift;
    my $referred_from = shift;

    my $language = $oCookie->getElement('language');
    my $partner = $oCookie->getElement('partner');

    if ($language eq 'spanish') {
        my $text = 'un amigo que usted refirió';
        if ($referred_from eq '2') {
            $text = 'un usted compartió un fishero con';
        }
    }
    else {
        my $text = 'referred';
        if ($referred_from eq '2') {
            $text = 'shared a file with';
        }
    }

    my $text = 'referred';
    if ($referred_from eq '2') {
        $text = 'shared a file with';
    }

    ##comes in as k, change to megabytes
    my $mbs = $additional_quota/1024;

    my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef,
$oDBO);
    $oUserData->loadWhere("SEQ", $user_seq);
    my $email_address = $oUserData->fetchColumn("EMAIL_ADDRESS");
    my $name_first = $oUserData->fetchColumn("NAME_FIRST");
    my $name_last = $oUserData->fetchColumn("NAME_LAST");

    my $oTemplate = new XDrive::Template( {'language' => $language,
'partner_code' => $partner} );

    $oTemplate->load('received_5MB_tellafriend.shtml');

```

```
$oTemplate->tags( {'mbs' => $mbs,
                  'text' => $text} );
$oTemplate->clear();

my $message = $oTemplate->get;

my %toXdrive =
(
    To      => "$name_first $name_last <$email_address>",
    Bcc     => '',
    From    => "support\@xdrive.com",
    Message => $message,
    Subject => "Congratulations!"
);

sendmail(%toXdrive);

$oUserData->finish();
}
```

###signup_form.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999. Updated
## Fri Apr 5, 1996 to use new templates. Updated Wed Apr 21 1999 to use
## new library code.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
```

```
use CGI;
use CGI::Carp 'fatalsToBrowser';
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Registration;
use XDrive::Template;
use XDrive::DatabaseO::Search;
use XDrive::Library;
```

```
use constant XD_REGISTRATION_DEFAULT_COUNTRY => 223;
```

```
exit &main;
```

```
sub main {
    my $oContent      = new XDrive::Template;
    my $oNavigation    = new XDrive::Template;
    my $oLayout        = new XDrive::Template;
    my $oCGI           = new CGI;

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    my $oSearch;

    my $sReferee      = $oCGI->param('referee');
    my $sClaimTicket  = $oCookie->getElement('ct');

    ## Defaults
    my $sUsername      = undef;
    my $sNameFirst     = undef;
    my $sNameLast      = undef;
    my $nYOB           = undef;
    my $nPromoSeq      = undef;
    my $nGender         = 3;
    my $sEmailAddress  = undef;

    my ($country_seq, $occupation_seq, $postal_code, $ct_promo_seq);

    my %pullDownHash;
    if (XDDDBConnectionCheck() && XDNFSCheck())
    {
        $oSearch = XDrive::DatabaseO::Search->new(undef);
    }
    else
    {
        $sClaimTicket = undef;
        $oSearch = undef;
        %pullDownHash = generate_db_array();
    }
    if ($sClaimTicket) {
        my $rhData = getUserData($oSearch, $sClaimTicket);
```

```

    if ($rhData) {
        my $oNewCgi = CGI->new($rhData);

        $sUsername      = $oNewCgi->param('username');
        $sNameFirst      = $oNewCgi->param('name_first');
        $sNameLast       = $oNewCgi->param('name_last');
        $sEmailAddress   = $oNewCgi->param('email_address');
        $nYOB            = $oNewCgi->param('birth_year');
        $nGender         = $oNewCgi->param('gender');
        $occupation_seq  = $oNewCgi->param('occupation_seq');
        $country_seq     = $oNewCgi->param('country_seq');
        $postal_code     = $oNewCgi->param('postal_code');
    }
}

if ($sReferee ne "") {
    # my $oCookie = XDrive::CGI::Cookie->new('x_session_info',
    $oCGI);
        my $sReferred_from = $oCGI->param('type');
        $oCookie->setElement({'partner_code'=>'xdrv'});
        $oCookie->setElement({'language'=>'english'});
        $oCookie->setElement({'referee' => $sReferee});
        $oCookie->setElement({'referred_from' => $sReferred_from});
        print "Set-Cookie: ".$oCookie->asString();
    }

    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');
    $oLayout->partner('xdrv');

    ## I'm assuming there will be one page and not a series of frames.
    ## this can be changed if need be
    # my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    # my $promo = $oCookie->getElement('promo');

    my $promo = $oCookie->getElement('promo');

    my $file_found;

    ##if we have a promo, try to get a special registration page
    if ($promo) {
        ##attempt to open a special registration page
        $file_found = $oLayout->load($promo . '_registration.shtml');
        if (!$file_found) {
            ##if we cannot, open the general promo reg page
            $file_found = $oLayout->load('promo_registration.shtml');
        }
    }

    ##is we don't have a promo then use the standard registration
    if ( (! $promo) || (! $file_found) ) {
        ## Load the required template HTML files.
        $oNavigation->load("front_nav.shtml");
        $oContent->load("front_signup.shtml");
        $oLayout->load("layout.shtml");

        $oContent->tags
        (
            'username'      => $sUsername,
            'name_first'    => $sNameFirst,
            'name_last'     => $sNameLast,
            'email_address' => $sEmailAddress,
        )
    }
}

```

```

        'country' =>
xd_form_countries_db_check(XD_REGISTRATION_DEFAULT_COUNTRY,
$oSearCh,\%pullDownHash),
        'occupation' => xd_form_occupation_db_check(undef,
$oSearCh,\%pullDownHash),
        'media_type' => xd_form_media_type_db_check(undef,
$oSearCh,\%pullDownHash),
        'gender' => xd_form_gender_db_check(undef,
$oSearCh,\%pullDownHash),
        'select_marketing' => 'CHECKED',
        'select_newsletter' => 'CHECKED',
        'referee' => $sReferee,
    ));

## Print out the HTML and exit
$oSearCh->tags
    (
        'header_graphic' => 'header_registration.gif',
        'title' => 'Register Now!',
        'content' => $oSearCh->get,
        'navigation' => $oSearCh->get
    );
}
elseif ($sClaimTicket) {
    $oSearCh->tags
        (
            'country' => xd_form_countries($country_seq,
$oSearCh),
            'occupation' => xd_form_occupation($occupation_seq,
$oSearCh),
            'media_type' => xd_form_media_type(undef, $oSearCh),
            'gender' => xd_form_gender($nGender, $oSearCh),
            'select_marketing' => 'CHECKED',
            'select_newsletter' => 'CHECKED',
            'username' => $sUsername,
            'name_first' => $sNameFirst,
            'name_last' => $sNameLast,
            'email_address' => $sEmailAddress,
            'birth_year' => $nYOB,
            'referee' => $sReferee,
            'postal_code' => $postal_code
        );
}
else {
    $oSearCh->tags
        (
            'country' =>
xd_form_countries_db_check(XD_REGISTRATION_DEFAULT_COUNTRY,
$oSearCh,\%pullDownHash),
            'occupation' => xd_form_occupation_db_check(undef,
$oSearCh,\%pullDownHash),
            'media_type' => xd_form_media_type_db_check(undef,
$oSearCh,\%pullDownHash),
            'gender' => xd_form_gender_db_check(undef,
$oSearCh,\%pullDownHash),
            'select_marketing' => 'CHECKED',
            'select_newsletter' => 'CHECKED',
            'referee' => $oCGI->param('referee'),
        );
}

$oSearCh->clear;

```

```

        print $oCGI->header, $oLayout->get;
        if (defined $oSearch)
        {
            $oSearch->disconnect();
        }

        return 0;
    }

## johngaa add to check of db is up or down
sub generate_db_array
{
    ## create a hash
    my %tempHash;
    my $i = 1;
    my $key;
    my @tempVal;
    open FH, "<down_data.dat";

    while(<FH>)
    {
        chomp $_;
        if ($_ =~ /^#(\w+)/g)
        {
            my @newArray;
            $i = 1;
            $key = $1;
            $tempHash{$key} = [ @newArray ];
        }
        else
        {
            @tempVal = split(/\~/, $_);

            $tempHash{$key}->[$i - 1][0] = $tempVal[0];
            $tempHash{$key}->[$i - 1][1] = $tempVal[1];
            $i++;
        }
    }

    close FH;
    return %tempHash;
}

sub xd_form_countries_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_countries(XD_REGISTRATION_DEFAULT_COUNTRY,
        $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $temp1 = $pullDownHash->{'country'};
        $returnVal = options_list(XD_REGISTRATION_DEFAULT_COUNTRY, @$temp1);
    }
}

```



```

    return $returnVal;
}

sub xd_form_occupation_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_occupation(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'occupation'};
        $returnVal = options_list(undef,@$templ);
    }

    return $returnVal;
}

sub xd_form_media_type_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_media_type(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'media_type'};
        $returnVal = options_list(undef,@$templ);
    }

    return $returnVal;
}

sub xd_form_gender_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_gender(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'gender'};
        $returnVal = options_list(undef,@$templ);
    }
}

```

```

    return $returnVal;
}

## end of johngaa add
sub getPromoURI ($$) {
    my $oSearch = shift;
    my @promo_seq = (shift);

    my $oDBH = $oSearch->fetchDBO->fetchDBH();

    my $st = "SELECT uri FROM xdrive.promo WHERE seq = ?";

    my $data = $oDBH->selectcol_arrayref($st, undef, @promo_seq);

    return $data->[0];
}

sub getUserData {
    my $oSearch = shift;
    my $sTicket = shift;

    my $oDBH = $oSearch->fetchDBO->fetchDBH();
    my $sQuery = "SELECT DATA FROM BATCH_USER_DATA WHERE CODE = ?";
    my $oCursor = $oDBH->prepare($sQuery);
    $oCursor->bind_param(1, $sTicket);
    $oCursor->execute;

    my $rh;
    my $sData = $oCursor->fetchrow_array();
    # my ($sData) = $oCursor->fetchrow_array();
    # eval $sData;
    # return $rh;
    return $sData;
}

```

###signup_success.cgi

```
#!/usr/bin/perl
## This CGI allows us to pass the sst and sid on to the inner frame
##
## Modified by Justin White on 10/14/99 by manually printing the
## header to the browser and getting rid of the XDrive::CGI import.
## Created new cgi, database, and error objects to pass to xd_security_check.
## Also added the exit in the sub call.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI::Carp qw(fatalsToBrowser);
use CGI ();
use Token;
use XDrive::Client::Security;
use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::Library;
use XDrive::CGI::Cookie;

&main();

exit;

sub main
{
    my $oCGI      = new CGI;
    my $oDBO      = new XDrive::DatabaseO;
    my $oErr      = new XDrive::Error;
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    #####
    ## Attempt to autenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If the authentication fails or there is an error during the
    ## authentication phase then redirect to the error CGI
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    #####
    ## Otherwise we have a valid session
    #####

    my $sUsername = $oToken->data('user');

    ### Edited by Justin so that the partner_code is looked for in
    ### the cookie instead of the token table.
    my $sPartner = $oToken->data('partner_code');
    my $sPartner = $oCookie->getElement('partner');
}
```

```
if (! defined $sPartner)
{
    $sPartner = "xdrv";
    $oCookie->setElement({'partner'=>$sPartner});
    print "Set-Cookie: ", $oCookie->asString();
}

my $oTemplate = new XDrive::Template( {'partner_code' => $sPartner}
);

$oTemplate->load('signup_success.thtml');
$oTemplate->tags( {'username' => $sUsername} );

print "content-type: text/html\n\n";

print $oTemplate->get();

$oDBO->disconnect();

return 0;
}
```

###signup_toc.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999. Updated
## Fri Apr 5, 1996 to use new templates.
##
## Modified by Justin White on 10/11/1999 so that it sets a cookie.
##
## Modified by Martin Hald on 11/15/1999 so that is now accepts
## - partner
## - language
## - agreeuri
## - disagreeuri

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::Template;
use XDrive::CGI::Cookie;

&main();

exit;

sub main {
    my $cookie;
    my $sPartnerCode;

    my $oCGI      = new CGI;
    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $sReferee = $oCGI->param('referee');
    my $sPartner = $oCGI->param('partner');
    my $sLanguage = $oCGI->param('language');
    my $sReferred_from = $oCGI->param('type');

    $oCookie->setElement({'partner_code'=>$sPartner});
    $oCookie->setElement({'language'=>$sLanguage});

    if ($sReferee ne "") {
        $oCookie->setElement({'referee' => $sReferee});
        $oCookie->setElement({'referred_from' => $sReferred_from});
        print "Set-Cookie: ".$oCookie->asString();
    }

    if (! defined $sPartner) {
        $sPartner = 'xdrv';
    }

    ## Load the terms and conditions
    my $hDefaults = {'partner_code'=>$sPartner, 'cookie'=>$oCookie};
    my $oContent  = new XDrive::Template($hDefaults);
    my $oLayout   = new XDrive::Template($hDefaults);

    $oContent->load('presignup.shtml');

    if ($sPartner eq 'xdrv') {
        my $oNavigation = new XDrive::Template($hDefaults);
        my $oHeader     = new XDrive::Template($hDefaults);
        my $oFooter     = new XDrive::Template($hDefaults);
    }
}
```

```

        $oLayout->load('layout.shtml');
        $oNavigation->load('front_nav.shtml');
        $oHeader->load('presignup_header.shtml');
        $oFooter->load('presignup_footer.shtml');
        $oContent->tags({'header' => $oHeader->get,
                        'footer'  => $oFooter->get, });
        $oLayout->tags({'navigation' => $oNavigation->get,
                        'header_graphic' => 'header_registration.gif',});
    } else {
        $oLayout->load('tac_wrapper.shtml');
    }

    my $sAgreeURI = $oCGI->param('agreeuri');
    my $sDisagreeURI = $oCGI->param('disagreeuri');

    $oLayout->tags({'title' => 'Terms and Conditions',
                  'content' => $oContent->get,
                  'agreeuri' => $sAgreeURI,
                  'disagreeuri' => $sDisagreeURI,});
    $oLayout->clear;

    print $oCGI->header();
    print $oLayout->get;

    return 0;
}

```

###skip_the_download.cgi

```
#!/usr/bin/perl

use strict;
use lib $ENV{PERL_XDRIVE_LIB};

use CGI qw(param redirect header cookie);
use CGI::Cookie;

use LWP::UserAgent;

use CGI::Carp qw(fatalsToBrowser);
use XDrive::Client::Security;
use XDrive::Client::Actions;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Search;
use XDrive::DatabaseO::Transaction;
use XDrive::Template;
use XDrive::CGI qw(:MAIN);
use XDrive::CGI::Cookie;
use XDrive::DatabaseO;
use XDrive::Error;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;
use Token;

my $oDBO = new XDrive::DatabaseO;
main($oDBO);

$oDBO->disconnect;
exit;

#####
## NOTE: Remove the quota check from here. will be handled in java.
#####

sub main
{
    my $oDBO = shift;
    my $oCGI = CGI->new();
    my $oErr = new XDrive::Error;
    my $oCookie = XDrive::CGI::Cookie->new('xd_std_info', $oCGI);

    ## params for file url and file name
    my $sFileURL = $oCGI->param('FILEURL');
    my $sFileName = $oCGI->param('FILENAME');
    my $sAltURL = $oCGI->param('ALTURL');
    my $sSid = $oCGI->param('SID');
    my $sGid = $oCGI->param('GID');
    my $sCatId = $oCGI->param('CATID');
    my $sPartnerCode = $oCGI->param('STDPartner');
    my $sLanguageCode = $oCGI->param('LANG');
    my $sUsername = $oCGI->param('user');
    my $sPassword = $oCGI->param('pass');
    my $sError = $oCGI->param('error');
    my $sCookie = $oCGI->cookie('SST');
```

```

my $sessionCookie;
my $sPromo = '';
my $sPartnerParams = "";
my $sCNetString = "";

## IF THE SPECIAL C|NET VARIABLES ARE DECLARED
## THEN GENERATE THE C|NET STRING
## THIS URL IS CALLED FOR ANY FILE DOWNLOADED
## FROM C|NET SO THAT THEY CAN CREDIT THE FILE
## BEING DOWNLOADED
if (
    ($sSid != '') &&
    ($sGid != '') &&
    ($sCatId != '')
) {

    $sAltURL = "http://beta.cnet.com/downloads/0-" . $sCatId . "-107-"
. $sSid . ".html?tag=ex.dl.xdrive";

    ## IF YOU ARE ON THE TEST SERVERS,
    ## THEN USE C|NET'S TEST URL
    if (
        ($ENV{'HTTP_HOST'} eq 'martini.xdrive.com') ||
        ($ENV{'HTTP_HOST'} eq 'antifreeze.xdrive.com')
    ){

        $sCNetString = "http://abv-sjc2-
export2.cnet.com/downloads/0,10152,0-" .
            $sCatId .
            "-110-" .
            $sSid .
            ",00.html?gid=" .
            $sGid .
            "&tag=ex.dl.xdrivepop.dlcgi." .
            $sSid;

        ## ELSE, USE THEIR REAL URL
    } else {

        $sCNetString = "http://abv-sjc1-
export2.cnet.com/downloads/0,10152,0-" .
            $sCatId .
            "-110-" .
            $sSid .
            ",00.html?gid=" .
            $sGid .
            "&tag=ex.dl.xdrivepop.dlcgi." .
            $sSid;

    }

}

$sPartnerParams =
"STDPARTNER=$sPartnerCode&LANG=$sLanguageCode&ALTURL=$sAltURL";

$soCookie->setElement(
    {
        'FILEURL'    => $sFileURL,
        'FILENAME'   => $sFileName,
    }
)

```



```

        'ALTURL'      => $sAltURL,
        'STDPARTNER'  => $sPartnerCode,
        'LANG'        => $sLanguageCode,
        'CATID'       => $sCatId,
        'SID'         => $sSid,
        'GID'         => $sGid,
    });

    print "Set-Cookie: ". $oCookie->asString();

    my $n = 0;
    my $rv;

    ## Create the database object
    my $oSearch = XDrive::Database0::Search->new($oDBO);

    ##The token for the user session
    my $oToken;

    ## If u/p
    if (defined $sUsername && defined $sPassword)
    {
        ## Auth or fail
        if (xd_auth_password($sUsername, $sPassword, $oDBO))
        {
            $oToken = xd_login($oCGI, $sUsername, $oErr);
            $sessionCookie = xd_set_session_cookie($oCGI,
            $sPartnerCode, $sLanguageCode, $sPromo);
        }
        else
        {
            ## Login failed
            my $r = getHTMLContent
            (
                'skip_the_download_login_failed.shtml',
                $sFileURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
            );

            print "Content-type: text/html\n\n";
            print $r;
            return 1;
        }
    }

    ## error or cookie not defined
    elsif ( (length($sError) > 0) || (length($sCookie) == 0) )
    {
        ## show the login page
        my $r = getHTMLContent('skip_the_download_login.shtml',
                                $sFileURL,
                                $sFileName,
                                $sAltURL,
                                $sPartnerCode,
                                $sLanguageCode
                            );

        print "Content-type: text/html\n\n";
        print $r;
    }

```

```

        return 1;
    }
    else
        ## cookie defined so authenticate it
        {
            $oToken = xd_security_check($oDBO,$oCGI,$oErr);
            $sessionCookie = xd_set_session_cookie($oCGI, $sPartnerCode,
            $sLanguageCode, $sPromo);

            if ($oErr->Occurud)
            {
                print $oCGI->redirect("/cgi-
bin/skip_the_download.cgi?&error=expired&$sPartnerParams");
                return 1;
            }
        }

        if (!$sFileURL) {
            my $thtml = ($sAltURL != '')?
'skip_the_download_no_alt_error.shtml'
                        : 'skip_the_download_error.shtml';

            my $sMessage = $oErr->ReturnMessageGivenCode(1220);

            &ThtmlErrorOut($thtml,
                $sMessage,
                $sFileURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
            );
        }

        ## create the Actions object and download the file
        my $oAction = new XDrive::Client::Actions($oToken,$oCGI);

        ## set the filename and file url
        $oAction->STDFileName($sFileName);
        $oAction->STDURL($sFileURL);

        ## see if file exists. if yes, give em message
        my $bFileExists = $oAction->STDFileExists();

        if ($bFileExists)
        {
            $oDBO->disconnect();
            my $sMessage = $oErr->ReturnMessageGivenCode(1242);

            ErrorOut($sMessage,$sFileURL,$sFileName,$sAltURL,$sPartnerCode,$sLangua
geCode);
        }

        ## Check that the file is not already being downloaded
        if ($oSearch->XDSTDBeingDownloaded($oToken->user,$sFileURL))
        {
            $oDBO->disconnect();
            my $sMessage = $oErr->ReturnMessageGivenCode(1243);

```

```
ErrorOut ($sMessage, $sFileURL, $sFileName, $sAltURL, $sPartnerCode, $sLanguageCode);
```

```
}
```

```
## Spool the action to download the file
my $oTransaction = new XDrive::DatabaseO::Transaction($oDBO);
my $nSeq = $oTransaction->insertSkipTheDownload
```

```
(
    $oToken->user,
    $sFileName,
    $sFileURL,
    0,
    undef
);
```

```
$oTransaction->commit;
```

```
## Insert failed return an error
```

```
if ($nSeq < 0)
```

```
{
    $oDBO->disconnect();
    my $sMessage = $oErr->ReturnMessageGivenCode(1244);
```

```
ErrorOut ($sMessage, $sFileURL, $sFileName, $sAltURL, $sPartnerCode, $sLanguageCode);
```

```
}
```

```
## IF THE INSERT DIDN'T FAIL,
## AND THE SPECIAL C|NET URL ISN'T NULL
## THEN CREDIT C|NET
elsif ($sCNetString ne '')
{
```

```
    my $oUA = new LWP::UserAgent;
    $oUA->agent("XDriveSTD/0.1 " . $oUA->agent);
```

```
    # Create a request
```

```
    my $oRequest = new HTTP::Request GET => $sCNetString;
```

```
    # Pass request to the user agent and get a response back
    my $oResult = $oUA->request($oRequest);
```

```
}
```

```
print redirect("/cgi-bin/skip_the_download_status.cgi?seq=$nSeq&$sPartnerParams");
}
```

```
sub ErrorOut ()
```

```
{
```

```
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
```

```

my $html = &getHTMLContent('skip_the_download_no_alt_error.html',
                           $sFileURL,
                           $sFileName,
                           $sAltURL,
                           $sPartnerCode,
                           $sLanguageCode,
                           $sMessage,
                           );

print "Content-type: text/html\n\n";
print $html;
exit(0);
}

sub ThtmlErrorOut ()
{
    my $thtml = shift;
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    my $html = &getHTMLContent($thtml,
                               $sFileURL,
                               $sFileName,
                               $sAltURL,
                               $sPartnerCode,
                               $sLanguageCode,
                               $sMessage,
                               );

    print "Content-type: text/html\n\n";
    print $html;
    exit(0);
}

sub getHTMLContent
{
    my $thtmlfile = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sMessage = shift;

    my $template = new XDrive::Template
    (
        {
            'partner_code' => $sPartnerCode,
            'language' => $sLanguageCode,
            'file' => $thtmlfile,
            'tags' =>
            {
                'FILE_URL' => $sFileURL,
                'FILE_NAME' => $sFileName,
                'ALTURL' => $sAltURL,
                'LANG' => $sLanguageCode,
            }
        }
    );

```

```
'STDPartner' => $sPartnerCode,
'message' => $sMessage,
}
));

$template->clear();

return $template->get;
}

## Create a string which makes the previously created
## cookie expire.

sub empty_cookie
{
    my $oSelf = shift;
    my $cookie = new CGI::Cookie
    (
        -name      => 'sst',
        -value     => '',
        -expires   => '-1M'
    );
    print header(-cookie=>[$cookie]);
}
```

###skip_the_download_status.cgi

```

#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header redirect);
use XDrive::CGI;
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::SkipDownload;
use XDrive::Template;
use XDrive::Error;
use XDrive::Library;
use Token;

use strict;

use constant TEMP_DIR => XDSTDTempDirectory();

&main;
exit(0);

sub main
{
    ## get parameters
    my $nFileSize;
    my $sTempFile;
    my $sFileName;
    my $sError;
    my $nStatus;
    my $bDone;
    my $percent = 0;
    my $nDownloadedSize = 0;
    my $sURL;
    my $nNow;

    my $oCGI = new CGI();
    my $nSeq = $oCGI->param('seq');
    my $nStart = $oCGI->param('start');
    my $sPartnerCode = $oCGI->param('STDPARTNER');
    my $sLanguageCode = $oCGI->param('LANG');
    my $sAltURL = $oCGI->param('ALTURL');
    my $previous_percent = $oCGI->param('pp');

    ## SET THE CONNECTION_COUNT = 0 IF IT ISN'T PASSED IN
    my $connection_count = ($oCGI->param('cc')) ? $oCGI->param('cc') : 0;

    my $oErr = new XDrive::Error;

    ## get the token and the action object
    my $oDBO = new XDrive::DatabaseO;
    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);
    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);

    my $sPartnerParams =
"STDPARTNER=$sPartnerCode&LANG=$sLanguageCode&ALTURL=$sAltURL";

    if ($oErr->Occurud)

```

```

{
    print redirect("/cgi-bin/skip_the_download.cgi?${$sPartnerParams}");
    return;
}

## if the sequence number was passed then get information from the
database.
if (defined $nSeq)
{
    ## load the information from the database
    my $oSkip = XDrive::DatabaseO::Table::SkipDownload->new(undef, $oDBO);
    $oSkip->loadWhere('SEQ',$nSeq);
    $nFileSize = $oSkip->fetchColumn('FILE_SIZE_BYTES');
    $sTempFile = $oSkip->fetchColumn('FILENAME_FOR_TEMP_FILE');
    $sFileName = $oSkip->fetchColumn('FILE_NAME');
    $nStatus = $oSkip->fetchColumn('IS_ACTIVE');
    $sError = $oSkip->fetchColumn('ERROR_CODE');
    $sURL = $oSkip->fetchColumn('FILE_URL');
    $bDone = $oSkip->fetchColumn('IS_DONE');
}

## XDRIVE.SKIP_THE_DOWNLOAD.IS_ACTIVE llegend
## 0 - still in queue
## 1 - being downloaded
## 2 - on hold

## IF CONNECTION_COUTN > 9, THEN GO TO THE FILE NOT FOUND (1220) ERROR
## DISPLAY, BUT KEEP TRYING TO DOWNLOAD THE FILE
if ($connection_count > 9) {
    $sError=1220;
}

## IF AN ERROR OCCURRED THEN DISPLAY IT
## AND THEN EXIT(0);
if (defined $sError)
{
    if ($sError == 1240)
    {
        &DisplayQuotaError('',
            $sURL,
            $sFileName,
            $sAltURL,
            $sPartnerCode,
            $sLanguageCode
        );
    }
    else
    {
        my $oErr = new XDrive::Error;
        $oErr->AddErrorByErrorCode($sError);
        &DisplayError($oErr->Message(),
            $sURL,
            $sFileName,
            $sAltURL,
            $sPartnerCode,
            $sLanguageCode
        );
    }
}

## IF THERE IS NO ERROR, THEN GATHER STATUS
## AND DISPLAY TO THE USER

```

```

else
{

## Get file size, later change to get from a tmp file
my $sPath = TEMP_DIR."/$$TempFile";

## IF STATUS IS LISTED AS DONE IN THE DB,
## THEN SHOW THE DONE PAGE
if ($bDone == 1)
{
    &DisplayDone('',
                $sURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
    );
}

## ELSE FILE IS NOT DONE,
## GATHER MORE DATA AND DISPLAY TO USER
else
{

## IF STATUS IS NOT ACTIVE, OR THE FILE DOESN'T EXIST
## THEN DISPLAY THE CONTACTING SERVER PAGE
## REMOVED: || ! -e $sPath
## FROM CHECK
if ( ($nStatus == 0 || -e $sPath)
    &&(!($previous_percent >= 0))
    )
{
    &DisplayContactServer($nSeq,$sURL,$sFileName,$sAltURL,$sPartnerCode,$sLanguageCode,$sPartnerParams,$connection_count);
}

## ELSE, GATHER STATUS DATA
## AND DISPLAY TO USER
else
{

## Set the start time in seconds since the epoch if not passed
## as parameter
if (! defined $nStart || $nStart !~ /^\\d+$/)
{
    $nStart = time();
}

## IF NO FILE SIZE HAS BEEN SET IN THE DB
## DISPLAY ZERO PERCENTAGES TO THE USER
if (! defined $nFileSize || $nFileSize == 0)
{
    $nFileSize = '0';
    $percent = '0';
    &DisplayStatus($nSeq,$percent,$sFileName,$nFileSize,'',
                  $nStart,'','',
    $sAltURL,$sPartnerCode,$sLanguageCode,$sPartnerParams);
}
}
}

```



```

    )

    ## ELSE
    ## * THERE WAS NO ERROR
    ## * THE FILE WAS NOT DONE
    ## * THE FILE EXISTS IN THE TEMPORARY DIRECTORY
    ## * THE DB HAS AN EXPECTED FILE SIZE
    ## SO READ THE FILE, CALCULATE DATA, AND DISPLAY TO USER
    else
    (

        ## These checks are performed before inserting the skip
        information    ## into the database, but we will do it again here to be
        safe.

        #   my $sError = $oErr->ReturnMessageGivenCode(141);
        #   XDErrorToBrowser("", $sError, undef, $oToken);
        ##die "Cannot check $sPath" if $sPath =~ /\.\.\/;
        ##die "Cannot check $sPath" if $sPath =~ /\.\.\/;

        ## Get the size of the download object
        my @file_info = stat($sPath);

        ## Conver the downloaded file size into KB
        if ($file_info[7] > 0)
        {
            $nDownloadedSize = $file_info[7];

            if ($nFileSize > 0)
            {
                $percent = 100 * $nDownloadedSize/$nFileSize;
            }
            if ($percent < 0)
            {
                $percent = 0;
            }
            $percent = sprintf("%.2f", $percent);
        }

        ## IF THE FILE IS GONE NOW, OR SOMEOTHER CONDITION, THE USER
        ## WILL NEVER SEE THE %DONE DROP
        ## USE WHICH EVER IS LARGER, THE PRECENT THAT WE JUST
        DISPLAYED    ## OF THE ONE THAT WE JUST READ FROM THE FILE SYSTEM
        $percent = ($previous_percent > $percent) ? $previous_percent
        : $percent;

        ## We have already transfered some of the file, so we can now
        ## estimate the download time.
        $nNow = time();

        my $sInfo;
        my $nElapsedSec = $nNow - $nStart;
        my $nTransPerSec = 0;

        if ($nElapsedSec)
        {
            $nTransPerSec = $file_info[7]/$nElapsedSec;
        }

        if ($nTransPerSec > 0)
    
```

```

    {
        my $partial = $percent/100;
        my ($nSecsRemain, $nMin, $nSecs, $nTransPerSecMB);

        if ($partial == 0) {
            $sInfo = '';
        } else {
            $nSecsRemain = ($nElapsedSec/$partial)-$nElapsedSec;
            $nMin = int($nSecsRemain/60);
            $nSecs = $nSecsRemain % 60;
            $nTransPerSecMB = $nTransPerSec/1024;
        }

        $sInfo = sprintf(", %d:%02d remaining (%.2f
KB/sec)", $nMin, $nSecs
                                , $nTransPerSecMB);
    }

    my $nTrans;

    my $k = "KB";
    my $nDiv = 1024;
    my $nTempSize = $file_info[7] || 0;

    if ($nFileSize > 1024*1024)
    {
        $k = "MB";
        $nDiv = 1024*1024;
    }

    if ($nFileSize < 0)
    {
        $nFileSize = 0;
    }

    $nFileSize = sprintf("%.2f", $nFileSize/$nDiv);
    $nTrans = sprintf("%.2f", $nTempSize/$nDiv);

    &DisplayStatus($nSeq, $percent, $sFileName, $nFileSize, '',
        $nStart, $sInfo, $k,
        $sAltURL, $sPartnerCode, $sLanguageCode, $sPartnerParams);

    ## END OF READING DATA FROM SYSTEM AND
    ## DISPLAYING TO USER
    }

    ## END OF NO EXPECTED SIZE IN DB
    ## SHOW USER ZERO PERCENTAGES
    }

    ## END OF FILE MUST BE DONE
    ## SO SHOW A DONE
    }

    ## END OF NO ERROR
    }

    $oDBO->disconnect;
}

```

```

sub DisplayContactServer
{
    my
    ($nSeq,$sURL,$sFileName,$sAltURL,$sPartnerCode,$sLanguageCode,$sPartnerParams
    , $connection_count) = @_;

    my ($sHostname) = $sURL =~ /\:\/\/([\^\/]+)\//;
    $connection_count++;

    ## load the status page
    my $template = new XDrive::Template
    (
        'partner_code' => $sPartnerCode,
        'language' => $sLanguageCode,
        'file' => 'skip_the_download_contacting.shtml',
        'tags' =>
        (
            'hostname' => $sHostname,
            'continue_to' => "/cgi-
bin/skip_the_download_status.cgi?seq=$nSeq&cc=$connection_count&$sPartnerPara
ms",
            'fileName' => $sFileName,
            'altURL' => $sAltURL,
        )
    );
    print "Content-type: text/html\n\n";
    print $template->get;
}

```

```

sub DisplayStatus
{
    my $nSeq = shift;
    my $percent = shift;
    my $filename = shift;
    my $filesize = shift;
    my $transferred = shift;
    my $start = shift;
    my $info = shift;
    my $k = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sPartnerParams = shift;

    my $percent_disp;

    if ($filesize <= 0)
    {
        $filesize = 'Unknown';
        $k = ' ';
        $percent_disp = 'Unknown';
        $percent = 0;
    }
    else
    {
        $percent_disp = "$percent%";
    }

    ## load the status page
    my $template = new XDrive::Template
    (

```

```

'partner_code' => $sPartnerCode,
'language' => $sLanguageCode,
'file' => 'skip_the_download_status.shtml',
'tags' =>
(
  'PERCENT_DISP' => $percent_disp,
  'PERCENT' => $percent,
  'FILE_NAME' => $filename,
  'FILE_SIZE' => $filesize,
  'TRANSFERRED' => $stransferred,
  'TRANSINFO' => $sinfo,
  'K' => $k,
  'URL' => "/cgi-
bin/skip_the_download_status.cgi?seq=$nSeq&start=$start&pp=$percent&$sPartner
Params",
  'altURL' => $sAltURL
)
));

$template->clear;
print "Content-type: text/html\n\n";
print $template->get;
}

```

```
sub DisplayDone
```

```

{
  my $sMessage = shift;
  my $sFileURL = shift;
  my $sFileName = shift;
  my $sAltURL = shift;
  my $sPartnerCode = shift;
  my $sLanguageCode = shift;

  &ErrorOut('skip_the_download_complete.shtml',
    $sFileURL,
    $sFileName,
    $sAltURL,
    $sPartnerCode,
    $sLanguageCode,
    $sMessage
  );
}

```

```
sub DisplayError
```

```

{
  my $sError = shift;
  my $sFileURL = shift;
  my $sFileName = shift;
  my $sAltURL = shift;
  my $sPartnerCode = shift;
  my $sLanguageCode = shift;

  my $html = ($sAltURL != '')? 'skip_the_download_no_alt_error.shtml'
    : 'skip_the_download_error.shtml';

  &ErrorOut($html,
    $sFileURL,
    $sFileName,
    $sAltURL,
    $sPartnerCode,

```

```

        $sLanguageCode,
        $sError
    );
}

```

```

sub DisplayQuotaError
{
    my $sError = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    &ErrorOut('skip_the_download_quota_error.shtml',
        $sFileURL,
        $sFileName,
        $sAltURL,
        $sPartnerCode,
        $sLanguageCode,
        $sError
    );
}

```

```

sub ErrorOut ()
{
    my $sHTMLFILE = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sMessage = shift;

    my $template = new XDrive::Template
    (
        {
            'language' => $sLanguageCode,
            'partner_code' => $sPartnerCode,
            'file' => $sHTMLFILE,
            'tags' =>
            (
                'message' => $sMessage,
                'altURL' => $sAltURL,
                'fileURL' => $sFileURL,
                'FILE_NAME' => $sFileName,
                'LANG' => $sLanguageCode,
                'ALTURL' => $sAltURL,
                'STDPARTNER' => $sPartnerCode,
            )
        }
    );

    my $html = $template->get;

    print "Content-type: text/html\n\n";
    print $html;
}

```



```

        $$LanguageCode,
        $$Error
    );
}

sub DisplayQuotaError
{
    my $$Error = shift;
    my $$FileURL = shift;
    my $$FileName = shift;
    my $$AltURL = shift;
    my $$PartnerCode = shift;
    my $$LanguageCode = shift;

    &ErrorOut('skip_the_download_quota_error.shtml',
        $$FileURL,
        $$FileName,
        $$AltURL,
        $$PartnerCode,
        $$LanguageCode,
        $$Error
    );
}

```

```

sub ErrorOut ()
{
    my $$THTMLFILE = shift;
    my $$FileURL = shift;
    my $$FileName = shift;
    my $$AltURL = shift;
    my $$PartnerCode = shift;
    my $$LanguageCode = shift;
    my $$Message = shift;

    my $template = new XDrive::Template
    (
        (
            'language' => $$LanguageCode,
            'partner_code' => $$PartnerCode,
            'file' => $$THTMLFILE,
            'tags' =>
            (
                'message' => $$Message,
                'altURL' => $$AltURL,
                'fileURL' => $$FileURL,
                'FILE_NAME' => $$FileName,
                'LANG' => $$LanguageCode,
                'ALTURL' => $$AltURL,
                'STDPARTNER' => $$PartnerCode,
            )
        )
    );

    my $html = $template->get;

    print "Content-type: text/html\n\n";
    print $html;
}

```

```

my $sUser_name = $oUserInfo->fetchColumn('NAME_FIRST') . " " .
$oUserInfo->fetchColumn('NAME_LAST');
my $sUser_email = $oUserInfo->fetchColumn('EMAIL_ADDRESS');
$oUserInfo->finish();
$oUserInfo->disconnect();

if ($sAddress)
{
    &send_mail($sName, $sAddress, $sUser_name, $sUser_email,
    $nUser_ID, $oCGI, $oToken, $oErr, $oCookie);
    &display_thank_you($oCGI,$oCookie);
}
else
{
    &display_form($oCGI,$oCookie);
}
}

sub send_mail {
    my ($sName, $sAddress, $sUser_name, $sUser_email, $nUser_ID, $oCGI,
    $oToken, $oErr, $oCookie) = @_;

    ## send out email for each friend only if form is filled out
    ## get number of friend fields

    my $numFriends = $oCGI->param("numFriends");
    for (my $i=1; $i<=$numFriends; $i++)
    {
        $sAddress = $oCGI->param('friends_email' . $i);
        $sName = $oCGI->param('friends_name' . $i);
        my $sMessage = &get_message($sUser_name, $nUser_ID, $sName,
        $sUser_name,$oCookie);

        ##only send the mail if the email address is filled out
        if ($sAddress)
        {
            my %toXdrive =
            (
                To      => "$sName <$sAddress>",
                Bcc      => "",
                From     => "$sUser_email",
                Message  => $sMessage,
                Subject  => "Check out X:drive!",
            );

            unless (sendmail %toXdrive)
            {
                warn "## Mail error ".$Mail::Sendmail::error;
                if ($Mail::Sendmail::error =~ /451/)
                {
                    my $sError = $oErr->ReturnMessageGivenCode(1310);
                    XDErrorToBrowser("", $sError, undef, $oToken);
                }
            }
        }
        else
        {
            my $sError = $oErr->ReturnMessageGivenCode(1311);
            XDErrorToBrowser('tell_a_friend__error.shtml', $sError, undef, $oToken);
        }
        exit(1);
    }
}

```


}

```

sub get_formfield {
    my ($sNum,$oCookie) = @_ ;

    my $oFormField = new XDrive::Template
        (
            'language'      => $oCookie->getElement('language'),
            'partner_code' => $oCookie->getElement('partner'),
        );
    $oFormField->load('tell_form_fields.shtml');

    $oFormField->tags
        (
            'number' => $sNum
        );

    return $oFormField->get;
}

sub get_message {
    my ($sUser_name, $nUser_ID, $sName, $sUserEmail,$oCookie) = @_ ;

    my $oMessage = new XDrive::Template
        (
            'language'      => $oCookie->getElement('language'),
            'partner_code' => $oCookie->getElement('partner'),
        );
    $oMessage->load('tell_a_friend_message.shtml');

    $oMessage->tags
        (
            'user_name' => $sUser_name,
            'nUser_ID' => $nUser_ID,
            'user_email' => $sUserEmail,
            'friend_name' => $sName
        );

    return $oMessage->get;
}

sub display_form {
    my $oCGI = shift;
    my $oCookie = shift;
    my $oForm = new XDrive::Template
        (
            'language'      => $oCookie->getElement('language'),
            'partner_code' => $oCookie->getElement('partner'),
        );
    $oForm->load('tell_a_friend.shtml');
    my $numFriends = $oCGI->param("numFriends");

    ##construct the html for multiple input fields
    my $inputFields='';

    for (my $i=1; $i<=$numFriends ; $i++)
    {
        $inputFields = $inputFields . &get_formfield($i,$oCookie);
    }
}

```

```
}

$soForm->tags
  (
    'friendsToTell' => $inputFields,
    'numFriends' => $numFriends,
  );
print $oCGI->header, $soForm->get;
exit(0);
}

sub display_thank_you {
  my $oCGI = shift;
  my $oCookie = shift;
  my $soForm = new XDrive::Template
    (
      'language' => $oCookie->getElement('language'),
      'partner_code' => $oCookie->getElement('partner'),
    );
  $soForm->load('tell_a_friend__t_y.t.html');
  print $oCGI->header, $soForm->get;
  exit(0);
}
```

###web_unauthorized.cgi

```

#!/usr/bin/perl
# Written by Martin Hald <mhald@uci.edu> on Sat Feb 13, 1999
#
# Program for showing unauthorized information and allowing the users to
# re-login and possibly showing them a "forgot your password?" link.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header param);
use CGI::Carp qw(fatalsToBrowser);
# use XDrive::CGI qw(:MAIN);
use XDrive::Client::Registration;
use XDrive::Template;
use XDrive::Error;

exit &main;

sub main
{
    my $oCGI = CGI->new();

    my $oLayout = new XDrive::Template;
    my $oContent = new XDrive::Template;
    my $oNavigation = new XDrive::Template;

    $oLayout->partner('xdrv');
    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');

    $oLayout->load('layout.thtml');
    $oNavigation->load('front_nav.thtml');

    ## Get the error key
    my $sError = $oCGI->param('error');

    ##now get the error message associated with that error
    my $oErr = new XDrive::Error;
    my $message = $oErr->ReturnMessageGivenCode($sError);

    ## Load the required template HTML files.
    my $oForm = new XDrive::Template;
    $oForm->partner('xdrv');
    $oForm->load("front_nav.thtml");
    $oContent->load("unauthorized.thtml");

    ## Update the layout
    $oLayout->tags
        (
            {
                'header_graphic' => 'header_denied.gif'
            }
        );

    ## Update the content
    $oContent->tags
        (
            {
                'error_message' => $message
            }
        );
    $oContent->clear();
}

```

```
## Print out the HTML and exit
$0Layout->tags
  ((
    'content' => $0Content->get,
    'navigation' => $0Navigation->get,
    'title' => 'Authorization Denied'
  ));
print header(), $0Layout->get;

return 0;
}
```

Windows Client Code

// Module: dlgShareAFile.h	1
// Module: dlgShareAFile.h	3
// Module: xdBase64.cpp	5
// Module: xdBase64.h	9
// Module: xdGlobals.h	10
// Module: xdParseDate.h	13
// Module: xdRegistry.h	14
// Module: xdTokens.h	16
// Module: xdTools.h	17
// Module: xdEngine.h	20
// Module: tdimsgtbl.h	22
// Module: tdisock.h	24
// Module: xdFileIO.cpp	41
// Module: xdDebugger.cpp	45

//

// **Module: dlgShareAFile.h**// Subsystem: KnowWare Internet Engine (kwEngine.dll)
// Contents: Declaration module for the dlgShareAFile class.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnowWare(r), Inc.

// All rights reserved.

//

// -----

//

#include "stdafx.h"

#include <xdGlobals.h>

#ifndef VXD_SOURCE_

#include "resource.h"

#endif

#include "dlgShareAFile.h"

#ifdef _DEBUG

#undef THIS_FILE

static char THIS_FILE[] = __FILE__;

#endif

// -----

// Implementation

//

BEGIN_MESSAGE_MAP(dlgShareAFile, CDialog)

//{{AFX_MSG_MAP(dlgShareAFile)

//}}AFX_MSG_MAP

END_MESSAGE_MAP()

// -----

// Method: dlgShareAFile()

// Purpose: Standard constructor

//

dlgShareAFile::dlgShareAFile(CWnd* pParent /*=NULL*/) : CDialog(dlgShareAFile::IDD, pParent)

{

//{{AFX_DATA_INIT(dlgShareAFile)

m_sFileName = szEMPTY;

m_sFileDescription = szEMPTY;

m_sEmailMessage = szEMPTY;

m_sEmailSubject = szEMPTY;

m_sEmail0 = szEMPTY;

m_sEmail1 = szEMPTY;

m_sEmail2 = szEMPTY;

m_sEmail3 = szEMPTY;

m_sEmail4 = szEMPTY;

//}}AFX_DATA_INIT

} // End of dlgShareAFile()

// -----

// Method: DoDataExchange()

// Purpose: Standard data exchange handler

//

void dlgShareAFile::DoDataExchange(CDataExchange* pDX)

{

```

CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(dlgShareAFile)
DDX_Text(pDX, IDC_SHARE_FILENAME, m_sFileName);
DDX_Text(pDX, IDC_SHARE_FILEDESC, m_sFileDescription);
DDX_Text(pDX, IDC_SHARE_EMAILMSG, m_sEmailMessage);
DDX_Text(pDX, IDC_SHARE_EMAILSUB, m_sEmailSubject);
DDX_Text(pDX, IDC_SHARE_EMAIL1, m_sEmail0);
DDX_Text(pDX, IDC_SHARE_EMAIL2, m_sEmail1);
DDX_Text(pDX, IDC_SHARE_EMAIL3, m_sEmail2);
DDX_Text(pDX, IDC_SHARE_EMAIL4, m_sEmail3);
DDX_Text(pDX, IDC_SHARE_EMAIL5, m_sEmail4);
//}}AFX_DATA_MAP
} // End of DoDataExchange()

// -----
// Method: OnInitDialog()
// Purpose: Called to initialize the contents of the dialog
//
BOOL dlgShareAFile::OnInitDialog()
{
    CDialog::OnInitDialog();

    UpdateData(FALSE);
    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
} // End of OnInitDialog()

// -----
// Method: OnOK()
// Purpose: Called to close out the dialog.
//
void dlgShareAFile::OnOK()
{
    UpdateData(TRUE);
    CDialog::OnOK();
} // End of OnOK()

```

//

// Module: dlgShareAFile.h

// Subsystem: KnoWare Internet Engine (kwEngine.dll)

// Contents: Declaration module for the dlgShareAFile class.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#if !defined(_INC_DLGSHAREAFILE_H_)

#define _INC_DLGSHAREAFILE_H_

#if _MSC_VER > 1000

#pragma once

#endif // _MSC_VER > 1000

#ifndef _VXD_SOURCE_

#include "resource.h"

#endif

#ifndef _VXD_SOURCE_

//

// -----

// dlgShareAFile dialog class

//

class dlgShareAFile : public CDialog

{

public:

dlgShareAFile(CWnd* pParent = NULL); // standard constructor

//{AFX_DATA(dlgShareAFile)

enum { IDD = IDD_SHARE };;

CString m_sFileName;

CString m_sFileDescription;

CString m_sEmailMessage;

CString m_sEmailSubject;

CString m_sEmail0;

CString m_sEmail1;

CString m_sEmail2;

CString m_sEmail3;

CString m_sEmail4;

//}}AFX_DATA

//{AFX_VIRTUAL(dlgShareAFile)

protected:

virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support

//}}AFX_VIRTUAL

protected:

//{AFX_MSG(dlgShareAFile)

virtual BOOL OnInitDialog();

virtual void OnOK();

//}}AFX_MSG

DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}

// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif

#endif // !defined(_INC_DLGSHAREAFILE_H_)

//

// Module: xdBase64.cpp

// Subsystem: X:drive Client Engine (xdEngine.dll)
 // Contents: Implementation module for the xdBase64 class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.

//

// -----

//

#include "stdafx.h"
 #include "xdBase64.h"

#ifdef _DEBUG
 #undef THIS_FILE
 static char THIS_FILE[] = __FILE__;
 #endif
 #ifdef _VXD_SOURCE_
 #include <xdEngine.h>
 #define TRACE_DEBUG_DPRINTF
 #endif

// Static Member Initializers
 //

// The 7-bit alphabet used to encode binary information
 CString xdBase64::m_sBase64Alphabet =
 _T("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/");

int xdBase64::m_nMask[] = { 0, 1, 3, 7, 15, 31, 63, 127, 255 };

//

// -----
 // Method: xdBase64()
 // Purpose: Standard Constructor

//

xdBase64::xdBase64 (void)
 {
 } // End of xdBase64()

//

// -----
 // Method: ~xdBase64()
 // Purpose: Standard destructor

//

xdBase64::~~xdBase64()
 {
 } // End of ~xdBase64()

//

// -----
 // Method: Encode()
 // Purpose: Encodes a string

//

CString xdBase64::Encode(LPCTSTR szEncoding, int nSize)
 {

 CString sOutput = _T("");
 int nNumBits = 6;
 UINT nDigit;
 int lp = 0;

```

    ASSERT( szEncoding != NULL );
    if( szEncoding == NULL )
        return sOutput;
    m_szInput = szEncoding;
    m_nInputSize = nSize;

    m_nBitsRemaining = 0;
    nDigit = read_bits( nNumBits, &nNumBits, lp );
    while( nNumBits > 0 )
    {
        sOutput += m_sBase64Alphabet[ (int)nDigit ];
        nDigit = read_bits( nNumBits, &nNumBits, lp );
    }
    // Pad with '=' as per RFC 1521
    while( sOutput.GetLength() % 4 != 0 )
    {
        sOutput += '=';
    }
    return sOutput;
} // End of Encode()

// -----
// Method: Decode()
// Purpose: Decodes data
// Notes: The size of the output buffer must not be less than 3/4 the
//        size of the input buffer. For simplicity, make them the same
//        size.
//
int xdBase64::Decode(LPCTSTR szDecoding, LPTSTR szOutput)
{
    CString sInput;
    int c, lp=0;
    int nDigit;
    CString strDecode;
    int* pDecode = (int*)strDecode.GetBuffer(256*sizeof(int));

    ASSERT( szDecoding != NULL );
    ASSERT( szOutput != NULL );
    if( szOutput == NULL )
        return 0;
    if( szDecoding == NULL )
        return 0;
    sInput = szDecoding;
    if( sInput.GetLength() == 0 )
        return 0;

    // Build Decode Table
    //
    for( int i = 0; i < 256; i++ )
        pDecode[i] = -2; // Illegal digit
    for( i=0; i < 64; i++ )
    {
        pDecode[ m_sBase64Alphabet[ i ] ] = i;
        pDecode[ m_sBase64Alphabet[ i ] | 0x80 ] = i; // Ignore 8th bit
        pDecode[ '=' ] = -1;
        pDecode[ '=' | 0x80 ] = -1; // Ignore MIME padding char
    }

    // Clear the output buffer
    memset( szOutput, 0, sInput.GetLength() + 1 );

    // Decode the Input

```

```

//
for( lp = 0, i = 0; lp < sInput.GetLength(); lp++ )
{
    c = sInput[ lp ];
    nDigit = pDecode[ c & 0x7F ];
    if( nDigit < -1 )
    {
        return 0;
    }
    else if( nDigit >= 0 )
        // i (index into output) is incremented by write_bits()
        write_bits( nDigit & 0x3F, 6, szOutput, i );
}
return i;
} // End of Decode()

// -----
// Method: read_bits()
// Purpose: dunno
//
UINT xdBase64::read_bits(int nNumBits, int * pBitsRead, int& lp)
{
    ULONG lScratch;
    while( ( m_nBitsRemaining < nNumBits ) &&
           ( lp < m_nInputSize ) )
    {
        int c = m_szInput[ lp++ ];
        m_lBitStorage <<= 8;
        m_lBitStorage |= (c & 0xff);
        m_nBitsRemaining += 8;
    }
    if( m_nBitsRemaining < nNumBits )
    {
        lScratch = m_lBitStorage << ( nNumBits - m_nBitsRemaining );
        *pBitsRead = m_nBitsRemaining;
        m_nBitsRemaining = 0;
    }
    else
    {
        lScratch = m_lBitStorage >> ( m_nBitsRemaining - nNumBits );
        *pBitsRead = nNumBits;
        m_nBitsRemaining -= nNumBits;
    }
    return (UINT)lScratch & m_nMask[nNumBits];
} // End of read_bits()

// -----
// Method: write_bits()
// Purpose: dunno
//
void xdBase64::write_bits ( UINT nBits, int nNumBits, LPTSTR szOutput, int& i )
{
    UINT nScratch;

    m_lBitStorage = (m_lBitStorage << nNumBits) | nBits;
    m_nBitsRemaining += nNumBits;
    while( m_nBitsRemaining > 7 )
    {
        nScratch = m_lBitStorage >> (m_nBitsRemaining - 8);
        szOutput[ i++ ] = (TCHAR)(nScratch & 0xFF);
        m_nBitsRemaining -= 8;
    }
}

```

WO 01/33381
} // End of write_bits()

PCT/US00/30536

//

// **Module: xdBase64.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: Declaration module for the xdBase64 class.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#if !defined(_INC_XDBASE64_H_)

#define _INC_XDBASE64_H_

#ifdef _VXD_SOURCE_

#include <xdCString.h>

#endif

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

// -----

// xdBase64 encoder class

//

class xdBase64

{

public:

xdBase64 (void);

virtual ~xdBase64 (void);

virtual int

Decode (LPCTSTR szDecoding, LPTSTR szOutput);

virtual CString Encode (LPCTSTR szEncoding, int nSize);

protected:

void

write_bits (UINT nBits, int nNumBts, LPTSTR szOutput, int& lp);

UINT

read_bits (int nNumBits, int* pBitsRead, int& lp);

protected:

int

m_nInputSize;

int

m_nBitsRemaining;

ULONG

m_lBitStorage;

LPCTSTR

m_szInput;

static int m_nMask[];

static CString m_sBase64Alphabet;

};

#endif // !defined(_INC_XDBASE64_H_)

// Module: xdGlobals.h

// Subsystem: X:drive

// Contents: Global definitions used throughout the system

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDGLOBALS_H_

#define _INC_XDGLOBALS_H_

#ifdef _VXD_SOURCE_

//

// This HodgePodge helps us to be able to compile all of our code

// under Ring-3 and Ring-0 without too much modification.

//

#ifndef USE_NDIS

#define USE_NDIS

#endif

#include <vtoolscp.h>

// VToolsD main header file

#ifndef LPCTSTR

typedef char TCHAR;

typedef unsigned char TCHAR;

typedef const TCHAR* LPCTSTR;

typedef TCHAR* LPTSTR;

typedef unsigned char BYTE;

typedef BYTE* LPBYTE;

typedef DSKTLSYSTEMTIME SYSTEMTIME;

typedef HANDLE HINSTANCE;

#define _T(x) (x)

#endif

#ifndef BASED_CODE

#define BASED_CODE

#endif

#ifndef INVALID_HANDLE_VALUE

#define INVALID_HANDLE_VALUE (HANDLE)-1

#endif

#define _tcsstr strstr // Standard unicode mappings

#define _tcslen strlen

#define _tscpy strcpy

#define _tcsrchr strrchr

#define _tscat strcat

#define _ttoi atoi

#define _ttol atol

#define _tcsrev strrev

#define _tcschr strchr

#define _tcsncpy strncpy

#define _tcsprbrk strpbrk

#define _stprintf sprintf

#define _tcslwr strlwr

```

#define _tcsupr      strupr
#define _tcsicmp     stricmp
#define _tscmp       strcmp
#define _tscoll      strcmp
#define _istdigit    isdigit
// #define ASSERT Assert
typedef HANDLE      HWND;
#endif

// -----
// Setup a whole bunch of constants that we can use throughout the systems
//
#define chNL          _T('\n')
#define chCOMMA       _T(',')
#define chDOSSLASH    _T('\\')
#define chUNIXSLASH   _T('/')
#define chQUOTE       _T('"')
#define chDQUOTE      _T("\'")
#define chPERIOD      _T('.')
#define chBAR         _T('|')
#define chTAB         _T('\t')
#define chCR          _T('\r')
#define chSPACE       _T(' ')
#define chCOLON       _T(':')
#define chSEMICOLON   _T(';')
#define chDASH        _T('-')
#define chPLUS        _T('+')
#define chPERCENT     _T('%')
#define chOPENBRACKET _T('[')
#define chCLOSEBRACKET _T(']')
#define chNUL         _T('\0')
#define chZERO        _T('0')
#define chONE         _T('1')
#define chTWO         _T('2')
#define chTHREE       _T('3')
#define chFOUR        _T('4')
#define chFIVE        _T('5')
#define chSIX         _T('6')
#define chSEVEN       _T('7')
#define chEIGHT       _T('8')
#define chNINE        _T('9')
#define chOPENPAREN    _T('(')
#define chCLOSEPAREN  _T(')')
#define chAT          _T('@')

#define szNL          _T("\n")
#define szCOMMA       _T(",")
#define szDOSSLASH    _T("\\")
#define szUNIXSLASH   _T("/")
#define szQUOTE       _T("\"")
#define szDQUOTE      _T("\'")
#define szPERIOD      _T(".")
#define szBAR         _T("|")
#define szTAB         _T("\t")
#define szCR          _T("\r")
#define szSPACE       _T(" ")
#define szCOLON       _T(":")
#define szSEMICOLON   _T(";")
#define szDASH        _T("-")
#define szPLUS        _T("+")
#define szOPENBRACKET _T("[")
#define szCLOSEBRACKET _T("]")

```



```

#define szAT _T("@")
#define szEMPTY _T("")
#define szCURRENTDIR _T(".")
#define szPARENTDIR _T("..")
#define szFTP_DOT _T("ftp.")
#define szFTP_SLASH _T("ftp://")
#define szOPENPAREN _T("(")
#define szCLOSEPAREN _T(")")

#define XD_CACHE_BASEDIR _T("xdcache")
#define XD_LOGFILE_NP _T("xdrive.log")
#define XD_LOGFILE_VXD _T("xdrivevxd.log")

// -----
// We need to define the scope of values which will be used in the system.
// They are defined here since we need to read/write these to the registry.
//

//
// General defines
//
#define XD_LEN_32 32
#define XD_LEN_64 64
#define XD_LEN_128 128
#define XD_LEN_256 256
#define XD_LEN_512 512
#define XD_LEN_1024 1024
#define XD_LEN_2048 2048

//
// these program IDs are also the 1st two digits of the registration number
//
#define XD_PROGID_XDRIVE 0x53 // {DB2112AD-0000-0000-0053-000004281965}

//
// IN will generate a directory listing and the local file that contains
// that information will have an extension of '.fnd'. For example, if
// IN/FND does a directory listing of ftp.microsoft.com/softlib/mslfiles,
// it will place the raw directory listing in the in the local IN cache
// directory (which is currently defined as hanging off of the same
// directory where IN is located) as
//
// c:\xdCache\ftp.microsoft.com\root.softlib.mslfiles.ls
//
// and the parsed FND formatted data will be placed into
//
// c:\xdCache\ftp.microsoft.com\root.softlib.mslfiles.fnd
//
// the .fnd file is parsed out to produce the information returned as a
// result of the FINDFIRST()/FINDNEXT() calls to the NP.
//
#define XD_FILEEXT_LS _T(".ls")
#define XD_FILEEXT_XDR _T(".fnd")
//
// Here is our Network Provider Name
//
#define XD_PROVIDER_NAME _T("Xdrive")
#define XD_PROVIDER_NETID 0x00120000

#endif // _INC_XDGLOBALS_H_

```

//

// **Module: xdParseDate.h**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Declaration module for the CParseDate utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnowWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDPARSEDATE_H_

#define _INC_XDPARSEDATE_H_

#include <xdTokens.h>

class XDTOOLS_PUBLIC CParseDate

{

public:

CParseDate (void);

~CParseDate (void);

BOOL Parse (LPCTSTR s);

int m_iYear;

int m_iMonth;

int m_iDay;

int m_iHour;

int m_iMinute;

int m_iSecond;

TCHAR m_szDate[64];

TCHAR m_szTime[32];

TCHAR m_szOrig[64];

private:

BOOL isNUM (LPCTSTR s);

BOOL isDOW (LPCTSTR s);

xdTokens m_tokens;

};

#endif

//

// **Module: xdRegistry.h**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Declaration module for the xdRegistry utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDREGISTRY_H_

#define _INC_XDREGISTRY_H_

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h> // X:drive Tools Related

//

// -----

// xdRegistry

// the registry class encapsulates the registry functions. You must open

// at least a hive in the constructor. then you can optionally open

// a subkey & read/write information to the registry. All methods will return

// true upon successful completion. false will be returned if an error

// has occurred.

//

class XDTOOLS_PUBLIC xdRegistry

{

public:

xdRegistry();

~xdRegistry();

//

// public interface

//

public:

BOOL RegOpenRead (HKEY hHive, LPCTSTR szSubKey);

BOOL RegOpenWrite (HKEY hHive, LPCTSTR szSubKey);

BOOL RegClose (void);

BOOL RegDeleteKey (HKEY hHive, LPCTSTR szSubKey);

BOOL RegDeleteValue (LPCTSTR szVal);

BOOL RegEnumKey (int i, LPCTSTR szKeyName, UINT uiLenWithNull);

BOOL RegEnumVal (int i, LPCTSTR szValName, UINT uiLenWithNull, LPCTSTR
szValData, UINT uiDataLenWithNull);

BOOL RegEnumStr (int i, LPCTSTR szVal, UINT uiLenWithNull);

BOOL RegGetStr (LPCTSTR sName, LPCTSTR szVal, UINT uiLenWithNull);

BOOL RegPutStr (LPCTSTR sName, LPCTSTR szVal);

BOOL RegPutBin (LPCTSTR sName, BYTE* pBuffer, UINT uiLen);

BOOL RegGetNum (LPCTSTR sName, BOOL& bVal);

BOOL RegGetNum (LPCTSTR sName, WORD& wVal);

BOOL RegGetNum (LPCTSTR sName, DWORD& dwVal);

BOOL RegGetNum (LPCTSTR sName, UINT& uiVal);

BOOL RegPutNum (LPCTSTR sName, DWORD dwVal);

```
LONG      RegGetLastError ( void );

private:
    HKEY    m_hKey;      // the current open hive
    LONG    m_lRetCode;  // the last return code
}; // End of xdRegistry

#endif // _INC_XDREGISTRY_H_
```

//

// **Module: xdTokens.h**// Subsystem: X:drive Tools Library (xdTools.dll)
// Contents: Declaration module for xdTokens utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.
// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
// All rights reserved.

//

// -----

//

#ifndef _INC_XDTOKENS_H_
#define _INC_XDTOKENS_H_#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000#include <xdGlobals.h> // X:drive system wide globals
#include <xdTools.h> // X:drive Tools Related

#define XD_MAX_TOKENS 1024

// -----

// xdTokens

// This class is a big worker class. its used to parse strings into
// tokens or substrings. Strings are parsed by supplying a string of
// characters which will be used to parse out the string.
//

class XDTOOLS_PUBLIC xdTokens

{

public:

xdTokens(LPCTSTR pTokens = NULL);
~xdTokens();

//

// Public Interface

//

public:

int Parse(int iNumToParse, LPCTSTR pString, LPCTSTR pTokens=NULL);
int Parse(LPCTSTR pString, LPCTSTR pTokens=NULL);
LPCTSTR operator[](int iIndex);

//

// Private Members

//

private:

LPCTSTR *m_pTok;
int m_iNumParsed;
LPTSTR m_szWorkString;
LPTSTR m_szTokens;
LPTSTR m_pWorkString;

}; // End of xdTokens

#endif // _INC_XDTOKENS_H_

//

// Module: xdTools.h

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Main header file for the xdTools library

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDTOOLS_H_

#define _INC_XDTOOLS_H_

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

#include <xdGlobals.h> // X:drive system wide globals

#ifdef VXD_SOURCE_

#include <xdCString.h>

#endif

#pragma warning (disable : 4100)

#pragma warning (disable : 4201)

//

// The following code block will insure the proper resolution of any

// API functions (and classes) which are exposed from the XDTOOLS library.

// When compiling the XDTOOLS library source code, make sure that the

// following #define is defined in the project settings (both debug & release).

// This will cause any classes and/or API functions defined as to

// be exported to the LIB file. If you are USING the library by linking to

// the XDTOOLS.LIB or XDTOOLS.D.LIB import libraries, then ignore the

// following #define's for

//

#ifdef _XDTOOLS_SOURCE_
#define XDTOOLS_PUBLIC __declspec(dllexport)

#else

#define XDTOOLS_PUBLIC // __declspec(dllimport)

#endif // _XDTOOLS_SOURCE_

//

// If we are debugging & we trap an exception, we will display it

// in a message box, otherwise in release mode, we wont.

//

#ifdef _DEBUG

#define XDTRACE(x) AfxMessageBox(x)

#else

#define XDTRACE(x) TRACE0(x)

#endif

// -----

// XDDATE API (Date Functions)

//

XDTOOLS_PUBLIC int XDDATE_MonthNum (LPTSTR szMonth);

// -----

// XDSTR API (String Functions)

//

```

XDTOOLS_PUBLIC LPTSTR XDSTR_Squish ( LPTSTR p );
XDTOOLS_PUBLIC LPTSTR XDSTR_StripChar ( LPTSTR p, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_DirSlashAdd ( LPTSTR sz, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_DirSlashRemove ( LPTSTR sz, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_TrimRight ( LPTSTR );
XDTOOLS_PUBLIC LPTSTR XDSTR_TrimLeft ( LPTSTR );
XDTOOLS_PUBLIC LPTSTR XDSTR_Trim ( LPTSTR );
XDTOOLS_PUBLIC BOOL XDAPI_CreatePath ( LPCTSTR ); // calls CreateDirectory() to make a path.

// -----
// Stuff for message boxes
//
#ifdef _VXD_SOURCE_
    int XDTOOLS_PUBLIC XD_MSG ( LPCTSTR szText, UINT uiMsgFlags );
    int XDTOOLS_PUBLIC XD_QUESTION ( LPCTSTR szText, UINT uiMsgFlags );
    LPCTSTR XDTOOLS_PUBLIC XD_TEXT ( HINSTANCE h, UINT uiResId ); // LOADS A
RESOURCE!
    BOOL XD_DoHelp ( LPHELPINFO );
    void XD_DoHelpContext ( CWnd* );
#endif

//
// the calling object needs to supply the resource
// handle for loading the string. So set up a stupid macro
// that will automatically supply this!
//
#define XD_LOADSTRING(x) XD_TEXT(AfxGetResourceHandle(),(x))

//
// DEBUGGING STUFF
//
#define CATCH_MSG_T("Caught Exception in File %s, Line %d\n\n")
#ifdef _VXD_SOURCE_
    #define XDCATCH dprintf(CATCH_MSG, T(__FILE__), __LINE__)
#else
    #define XDCATCH { CString s; s.Format(CATCH_MSG, T(__FILE__), __LINE__);
AfxMessageBox(s); }
#endif

//
// Ring 0 File I/O
//
#ifdef _VXD_SOURCE_
#define GENERIC_READ (0x80000000) /* from WINNT.H */
#define GENERIC_WRITE (0x40000000) /* from WINNT.H */
#define CREATE_NEW 1
#define CREATE_ALWAYS 2
#define OPEN_EXISTING 3
#define OPEN_ALWAYS 4
#define TRUNCATE_EXISTING 5
#define FILE_SHARE_READ 0x00000001
#define FILE_SHARE_WRITE 0x00000002
#define FILE_SHARE_DELETE 0x00000004 // not supported
#endif

HANDLE CreateFile ( LPCTSTR lpFileName, // pointer to name of the file
    DWORD dwDesiredAccess, // access (read-write) mode
    DWORD dwShareMode, // share mode
    void* lpSecAtt, // pointer to security
    attributes,
    DWORD dwCreateFlags, // how to create
    DWORD dwFlagsAndAttributes, // file attributes
    HANDLE);

```

```

BOOL CloseHandle ( HANDLE hFile );

```

```

BOOL ReadFile ( HANDLE hFile,          // handle of file to read
               void* lpBuffer,          // pointer to buffer that receives data
               DWORD nNumberOfBytesToRead, // number of bytes to read
               DWORD* lpNumberOfBytesRead, // pointer to number of bytes read
               void* lpOverlapped);      // pointer to structure for data

```

```

BOOL ReadFileLine ( HANDLE hFile,          // handle of file to read
                   BYTE* lpBuffer,          // pointer to buffer that receives
data                                     data
                   DWORD dwBytesToRead,      // number of bytes to read
                   DWORD* dwBytesRead,       // pointer to number of bytes read
                   DWORD* dwOffset);         // pointer to structure for data

```

```

BOOL WriteFile ( HANDLE hFile, LPCTSTR lpBuffer, DWORD dwBytesToWrite,
                DWORD* pBytesWritten, void* p);

```

```

DWORD      GetFileSize ( HANDLE hFile, DWORD* pdwHigh );
#endif

```

```

#endif // !defined(_INC_XDTOOLS_H_)

```


//

// **Module: xdEngine.h**// Subsystem: X:drive Client Engine (xdEngine.dll)
// Contents: Main include file for the xdEngine subsystem
//// -----
// Copyright (c) 1999 by X:drive(tm), Inc.
// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
// All rights reserved.
// -----
// -----
//#ifndef _INC_XDRIVE_ENGINE_H_
#define _INC_XDRIVE_ENGINE_H_#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000#pragma warning (disable : 4100)
#pragma warning (disable : 4201)#ifdef _XDENGINE_SOURCE_
#define XDAPI_PUBLIC __declspec(dllexport)
#else
#define XDAPI_PUBLIC // __declspec(dllimport)
#endif // _XDENGINE_SOURCE_

#pragma pack(1) // byte pack this thing!

#include <xdGlobals.h>

// -----
// XD_DIRENTRY - directory listing item
//// The following structure is used to hold an object in the file listing
// file. Xdrive will generate the file list for the directory and store it
// in the cache directory. That file will contain
// a list of record structures of this type. The .mnd file is generated
// based upon the FTP server specific format in the .idx file in the same
// cache directory.
//

typedef struct _xd_direntry_

{
 USHORT cb; // class size, MUST BE FIRST!!!!
 DWORD dwFileAttributes;
 FILETIME ftCreationTime;
 FILETIME ftLastAccessTime;
 FILETIME ftLastWriteTime;
 DWORD nFileSizeHigh;
 DWORD nFileSizeLow;
 TCHAR cFileName[XD_LEN_512];
 TCHAR m_szObPerms [XD_LEN_32 + 1];
 BYTE m_bObOwnerPerms[4];
 BYTE m_bObGroupPerms[4];
 BYTE m_bObWorldPerms[4];
} XD_DIRENTRY, * LPXD_DIRENTRY;

#pragma pack()

```

//
// Return codes
//
typedef UINT    XD_RETCODE;

#define XD_SUCCESS                (int)0
#define XD_CANCEL                 (int)1
#define XD_ERR_CONNECTFAILED     (int)2    // socket connect failed
#define XD_ERR_LOGINFAILED       (int)3    // bad username/pwd
#define XD_ERR_CONNECTREFUSED    (int)5    // socket connect refused
#define XD_ERR_CANTRESOLVEHOST   (int)6    // cant resolve host
#define XD_ERR_SERVERUPGRADING   (int)7    // upgrading our servers

#define XD_ERR_OTHER              (int)-1

//
// The following constants are used in the notification structure.
//
typedef enum
{
    XD_NOTIFY_IDLE                = 0,      // nothing happening here
    XD_NOTIFY_STATUS_MSG          = 1000,   // status msg
    XD_NOTIFY_XFERDATA_DN         = 1001,   // downloading
    XD_NOTIFY_XFERDATA_UP         = 1002,   // uploading
    XD_NOTIFY_QUOTA                = 1003,   // Update the quota
    XD_NOTIFY_START                = 1004,   // Start an operation
    XD_NOTIFY_STOP                = 1005     // Stop an operation
} XD_NOTIFY_CODE;

// -----
// XD_NOTIFY - This is our notification structure. The http engine
// will use this structure to pass status information back to the
// invoking method.
//
#pragma pack(1)

typedef struct _xd_notification_
{
    int                m_iNotifyType;
    TCHAR              m_szMessage [ 1024 + sizeof(TCHAR) ];

    //
    // used for send/receive
    //
    ULONG              m_dwStartTime;        // GetTickCount()/1000
    ULONG              m_dwCurrentTime;      // GetTickCount()/1000

    DWORD              m_dwCurrentBytes;
    DWORD              m_dwTotalBytes;

    TCHAR              m_szLocalFileName [ MAX_PATH + sizeof(TCHAR) ];
    TCHAR              m_szRemoteFileName [ MAX_PATH + sizeof(TCHAR) ];
} XD_NOTIFY, *LPXD_NOTIFY;
#pragma pack()
#define XD_NOTIFY_MAX    50

#endif // _INC_XDRIVE_ENGINE_H_

```

//

// **Module: tdimsgtbl.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: TDI Error table.

//

// -----
// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by River Front Software

// All rights reserved.

// -----

// -----

//

#ifndef __TDIMSGTBL_H

#define __TDIMSGTBL_H

typedef struct

```
{
    TDI_STATUS    Status;
    int           WinStatus;
    char          *szMsg;
} INETTDIMSG;
```

INETTDIMSG TdiMsgTbl[] =

```
{
    {TDI_SUCCESS, ERROR_SUCCESS, "TDI Success"},
    {TDI_NO_RESOURCES, ERROR_BAD_COMMAND, "No resources."},
    {TDI_ADDR_IN_USE, ERROR_BAD_COMMAND, "Address already in use."},
    {TDI_BAD_ADDR, ERROR_BAD_COMMAND, "Address given is bad."},
    {TDI_NO_FREE_ADDR, ERROR_BAD_COMMAND, "No addresses available."},
    {TDI_ADDR_INVALID, ERROR_BAD_COMMAND, "Address object is invalid."},
    {TDI_ADDR_DELETED, ERROR_BAD_COMMAND, "Address object was deleted."},
    {TDI_BUFFER_OVERFLOW, ERROR_BAD_COMMAND, "Buffer overflowed."},
    {TDI_BAD_EVENT_TYPE, ERROR_BAD_COMMAND, "Bad event type."},
    {TDI_BAD_OPTION, ERROR_BAD_COMMAND, "Bad option or length."},
    {TDI_CONN_REFUSED, ERROR_BAD_COMMAND, "Connection was refused."},
    {TDI_INVALID_CONNECTION, ERROR_BAD_COMMAND, "Invalid connection."},
    {TDI_ALREADY_ASSOCIATED, ERROR_BAD_COMMAND, "Connection already associated."},
    {TDI_NOT_ASSOCIATED, ERROR_BAD_COMMAND, "Connection not associated."},
    {TDI_CONNECTION_ACTIVE, ERROR_BAD_COMMAND, "Connection is still active."},
    {TDI_CONNECTION_ABORTED, ERROR_BAD_COMMAND, "Connection was aborted."},
    {TDI_CONNECTION_RESET, ERROR_BAD_COMMAND, "Connection was reset."},
    {TDI_TIMED_OUT, ERROR_BAD_COMMAND, "Connection timed out."},
    {TDI_GRACEFUL_DISC, ERROR_BAD_COMMAND, "Received a graceful disconnect."},
    {TDI_NOT_ACCEPTED, ERROR_BAD_COMMAND, "Data not accepted."},
    {TDI_MORE_PROCESSING, ERROR_BAD_COMMAND, "More processing required."},
    {TDI_INVALID_STATE, ERROR_BAD_COMMAND, "TCB in an invalid state."},
    {TDI_INVALID_PARAMETER, ERROR_BAD_COMMAND, "An invalid parameter."},
    {TDI_DEST_NET_UNREACH, ERROR_BAD_COMMAND, "Destination net is unreachable."},
    {TDI_DEST_HOST_UNREACH, ERROR_BAD_COMMAND, "Dest. host is unreachable."},
    {TDI_DEST_UNREACHABLE, ERROR_BAD_COMMAND, "Dest. is unreachable."},
    {TDI_DEST_PROT_UNREACH, ERROR_BAD_COMMAND, "Destination protocol is unreachable."},
    {TDI_DEST_PORT_UNREACH, ERROR_BAD_COMMAND, "Dest. port is unreachable."},
    {TDI_INVALID_QUERY, ERROR_BAD_COMMAND, "Invalid query type specified."},
    {TDI_REQ_ABORTED, ERROR_BAD_COMMAND, "Request was aborted for some reason."},
    {TDI_BUFFER_TOO_SMALL, ERROR_BAD_COMMAND, "Buffer was too small."},
    {TDI_CANCELLED, ERROR_BAD_COMMAND, "The request was cancelled."},
    {TDI_BUFFER_TOO_BIG, ERROR_BAD_COMMAND, "Invalid request."},
    {ERROR_SEM_TIMEOUT, ERROR_SEM_TIMEOUT, "Timed out."},
    {TDI_PENDING, ERROR_BAD_COMMAND, "Pending"}
}
```

WO 01/33381

PCT/US00/30536

};

#endif

//

// **Module: tdisock.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: TDI Socket header file.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by River Front Software

// All rights reserved.

// -----

// -----

//

#ifndef __TDISOCK_H

#define __TDISOCK_H

#define TDISOCK_TIMEOUT 15000

#define WSADESCRIPTION_LEN 256

#define WSASYS_STATUS_LEN 128

typedef short SHORT;

typedef unsigned short USHORT;

typedef unsigned short ushort;

typedef unsigned int uint;

typedef unsigned long ulong;

typedef unsigned long ULONG;

typedef void (*CTEReqCmpltRtn)(void *Context, long FinalStatus, unsigned int ByteCount);

typedef unsigned char uchar;

typedef struct WSADATA {

WORD wVersion;

WORD wHighVersion;

char szDescription[WSADESCRIPTION_LEN+1];

char szSystemStatus[WSASYS_STATUS_LEN+1];

unsigned short iMaxSockets;

unsigned short iMaxUdpDg;

char FAR * lpVendorInfo;

} WSADATA;

typedef WSADATA FAR *LPWSADATA;

#define USE_NDIS 1

#include <vtoolscp.h>

#include <crtl.h>

#undef USE_NDIS

#include <tdi.h>

#include <vxdsvc.h>

#include <tdivxd.h>

#include <tdistat.h>

#undef VTDI_Device_ID

#include <vtdi.h>

#define MAKELONG(a, b) (((LONG)((WORD)(a)) | ((DWORD)((WORD)(b))) << 16))

#define LOWORD(l) ((WORD)(l))

```

#define HIWORD(l)      (((WORD)(((DWORD)(l) >> 16) & 0xFFFF))
#define LOBYTE(w)      ((BYTE)(w))
#define HIBYTE(w)      ((BYTE)(((WORD)(w) >> 8) & 0xFF))

/*
 * Structures returned by network data base library, taken from the
 * BSD file netdb.h. All addresses are supplied in host order, and
 * returned in network order (suitable for use in system calls).
 */

struct hostent {
    char FAR * h_name;      /* official name of host */
    char FAR * FAR * h_aliases; /* alias list */
    short h_addrtype;      /* host address type */
    short h_length;        /* length of address */
    char FAR * FAR * h_addr_list; /* list of addresses */
#define h_addr h_addr_list[0] /* address, for backward compat */
};

/***** Wait for semaphore flags */
#define WAIT_SEMA_FLAGS 0 //BLOCK_SVC_INTS | BLOCK_POLL

/***** Macro to call wait on semaphore function */
#define SEMAPHORE_WAIT( hSem, nTimeout ) \
    WaitOnSemaphore( s, hSem, #hSem, nTimeout )

/***** Checks for valid TDI status */
#define TDI_CHECKSTATUS(s) if ( (s) != TDI_SUCCESS ) \
    { \
        \
        Line:%d TDI [%d] - %s\n", \
        MapTdiToString(s)); \
        \
        errdebug( DBG_log("ERROR - File: %s \
        __FILE__, __LINE__, (s), \
        goto Exit; \
    }

/***** Destroys a semaphore */
#define SEMAPHORE_SAFE_DESTROY(hSem) \
    if (hSem) \
    { \
        vbsdebug( DBG_log("Destroy Semaphore %s", #hSem); ); \
        UtilSemDestroy(hSem); \
        hSem = 0; \
    }

/***** Signals a semaphore */
#define SEMAPHORE_SAFE_SIGNAL(hSem) \
    if (hSem) \
    { \
        vbsdebug( DBG_log("*** Signal Semaphore %s", #hSem); ); \
        vbsdebug( DBG_log_hex_long( hSem ); ); \
        Signal_Semaphore_No_Switch( hSem ); \
    } \
    else \
    { \
        vbsdebug( DBG_log("*** NO SEMAPHORE TO SIGNAL %s", #hSem); ); \
    }

```

```

/*
 * Basic system type definitions, taken from the BSD file sys/types.h.
 */
typedef unsigned char  u_char;
typedef unsigned short u_short;
typedef unsigned int   u_int;
typedef unsigned long  u_long;

/*
 * Constants and structures defined by the internet system,
 * Per RFC 790, September 1981, taken from the BSD file netinet/in.h.
 */

/*
 * Protocols
 */
#define IPPROTO_IP      0      /* dummy for IP */
#define IPPROTO_ICMP    1      /* control message protocol */
#define IPPROTO_IGMP    2      /* internet group management protocol */
#define IPPROTO_GGP     3      /* gateway^2 (deprecated) */
#define IPPROTO_TCP     6      /* tcp */
#define IPPROTO_PUP     12     /* pup */
#define IPPROTO_UDP     17     /* user datagram protocol */
#define IPPROTO_IDP     22     /* xns idp */
#define IPPROTO_ND      77     /* UNOFFICIAL net disk proto */

#define IPPROTO_RAW     255     /* raw IP packet */
#define IPPROTO_MAX     256

/*
 * Port/socket numbers: network standard functions
 */
#define IPPORT_ECHO      7
#define IPPORT_DISCARD  9
#define IPPORT_SYSTAT   11
#define IPPORT_DAYTIME  13
#define IPPORT_NETSTAT  15
#define IPPORT_FTP       21
#define IPPORT_TELNET    23
#define IPPORT_SMTP      25
#define IPPORT_TIMESERVER 37
#define IPPORT_NAMESERVER 42
#define IPPORT_WHOIS     43
#define IPPORT_MTP       57

/*
 * Port/socket numbers: host specific functions
 */
#define IPPORT_TFTP      69
#define IPPORT_RJE       77
#define IPPORT_FINGER    79
#define IPPORT_TTYLINK   87
#define IPPORT_SUPDUP    95

/*
 * UNIX TCP sockets
 */
#define IPPORT_EXECSERVER 512

```

```

#define IPPORT_LOGINSERVER  513
#define IPPORT_CMDSERVER    514
#define IPPORT_EFSSERVER    520

/*
 * UNIX UDP sockets
 */
#define IPPORT_BIFFUDP      512
#define IPPORT_WHOSERVER    513
#define IPPORT_ROUTESERVER  520
/* 520+1 also used */

/*
 * Ports < IPPORT_RESERVED are reserved for
 * privileged processes (e.g. root).
 */
#define IPPORT_RESERVED    1024

/*
 * Link numbers
 */
#define IMPLINK_IP          155
#define IMPLINK_LOWEXPER    156
#define IMPLINK_HIGHEXPER   158

/*
 * Internet address (old style... should be updated)
 */
struct in_addr {
    union {
        struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b;
        struct { u_short s_w1,s_w2; } S_un_w;
        u_long S_addr;
    } S_un;
#define s_addr S_un.S_addr
/* can be used for most tcp & ip code */
#define s_host S_un.S_un_b.s_b2
/* host on imp */
#define s_net S_un.S_un_b.s_b1
/* network */
#define s_imp S_un.S_un_w.s_w2
/* imp */
#define s_impno S_un.S_un_b.s_b4
/* imp # */
#define s_lh S_un.S_un_b.s_b3
/* logical host */
};

#define htons(host) ( (((host) & 0xff) << 8) | ((host) >> 8) )
#define htonl( ULONG hostlong );

/*
 * Definitions of bits in internet address integers.
 * On subnets, the decomposition of addresses to host and net parts
 * is done according to subnet mask, not the masks here.
 */
#define IN_CLASSA(i)      (((long)(i) & 0x80000000) == 0)
#define IN_CLASSA_NET      0xff000000
#define IN_CLASSA_NSHIFT   24
#define IN_CLASSA_HOST      0x00ffffff
#define IN_CLASSA_MAX      128

```



```
#define IN_CLASSB(i)      (((long)(i) & 0xc0000000) == 0x80000000)
#define IN_CLASSB_NET      0xffff0000

// end first 30 pages aj
```

```
        int iMax = i;
        CString* pArray = new CString[iMax];
        i=0;
        while (r1.RegEnumKey(i++,szVal,dwCnt))
            pArray[i-1] = szVal;
        r1.RegClose();
        for (i=0; i<iMax; i++)
        {
            CString str = pArray[i];
            CString strTmp;
            strTmp.Format(_T("%s\\%s"), (LPCTSTR)szSubKey, (LPCTSTR)str);
            r1.RegDeleteKey(hHive,strTmp);
        }
        delete[] pArray;
    }

    //
    // then Delete the key
    //
    m_lRetCode = ::RegDeleteKey ( hHive, szSubKey );
#endif

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    //
    // bOK is TRUE if ERROR_SUCCESS was returned
    //
    bOK = (ERROR_SUCCESS == m_lRetCode);

    return bOK;
} // End of RegDelete()

// -----
// Method: RegClose()
// Purpose: the the registry is open, close it.
//
BOOL xdRegistry::RegClose ( )
{
    BOOL bOK = TRUE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

        if ( m_hKey != NULL )
            ::RegCloseKey ( m_hKey );
```

```

#ifndef _VXD_SOURCE_
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

//
// unconditionally null the key
//
m_hKey = NULL;

return bOK;
} // End of RegClose()

// -----
// Method: RegEnumStr()
// Purpose: enumerates subkeys for a key. i is the index to get
//
BOOL xdRegistry::RegEnumStr ( int i, LPCTSTR szValue, UINT uiLenWithNull )
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiLenWithNull;
    LPBYTE pValue = (LPBYTE) szValue;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifndef _VXD_SOURCE_
    try
    {
#endif
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiLenWithNull );

#ifndef _VXD_SOURCE_
        m_lRetCode = ::RegEnumKey ( m_hKey, dwIdx, // hive/key
                                   (LPTSTR)pValue, dwSize); // index
        // key
        // the size of the
    }
    catch (...)
    {
        m_lRetCode = ERROR_INVALID_PARAMETER;
    }
#endif

    //
    // buffer
    //
    #ifndef _VXD_SOURCE_
        CString sTmp;
        TCHAR szBuf = (BYTE*)sTmp.GetBuffer(512);
        m_lRetCode = ::RegEnumKeyA ( m_hKey, dwIdx, // hive/key
                                     (char*)buf, dwSize); // index of the
        // key name will
        // the size of the buffer
    #endif

    CString fred(buf);

```

```

        _tcscpy((LPTSTR)szValue,fred);
    #else
        m_IRetCode = ::RegEnumKey (    m_hKey,                // hive/key
                                     dwIdx,                    // index
                                     (LPTSTR)pValue,            // key
                                     dwSize);                  // the size of the
    of the key to get
    name will go here
    buffer
    #endif
#endif

    bOK = (ERROR_SUCCESS == m_IRetCode);
    if (bOK != FALSE)
    {
        //
        // terminate the string...ensure that we dont go past
        // the max lenh of the string!
        //
        ((LPTSTR)szValue) [ min(dwSize,uiLenWithNull) ] = 0;
    }

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegEnumStr()

// -----
// Method: RegGetStr()
// Purpose: retrieves a string value from the registry. NOTE: The length
//           of the string MUST include space for the NULL terminator since
//           this character IS read from the registry. So, if you want to
//           read 'ABCD' from the registry, supply a uiLenWithNull of five(5).
//
BOOL xdRegistry::RegGetStr ( LPCTSTR szName, LPCTSTR szValue, UINT uiLenWithNull )
{
    BOOL  bOK = TRUE;
    DWORD dwType = 0;
    DWORD dwSize  = (DWORD) uiLenWithNull;
    LPBYTE pValue  = (LPBYTE) szValue;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiLenWithNull );
    }
    catch (...)
    {
        //
    }
#endif
}

```

```

#ifdef _VXD_SOURCE_
    m_lRetCode = ::RegQueryValueEx (m_hKey,
        // value name
        // reserved
        // the REG_* type
        // pointer to the storage area
        // # to fetch (WITH NULL)
#else
    #ifdef _UNICODE
        char sShort[512];
        char sDefault[512];
        char buf[512];
        BOOL b;
        *sDefault = *sShort=0;
        WideCharToMultiByte (CP_ACP, 0, szName, -1, sShort, 512, sDefault, &b );
        m_lRetCode = ::RegQueryValueExA (m_hKey,
            // hive/key
            // value name
            // reserved
            // the REG_* type
            // pointer to the storage area
            // # to fetch (WITH NULL)
            CString fred(buf);
            _tcsncpy((LPTSTR)szValue,fred);
        #else
            m_lRetCode = ::RegQueryValueEx (m_hKey,
                // hive/key
                // value
                // reserved
                // the
                // pointer to the storage area
                // # to
                fetch (WITH NULL)
            #endif
        #endif

        bOK = (ERROR_SUCCESS == m_lRetCode);
        if ( bOK == TRUE )
        {
            //
            // make sure that it was a string value which was returned.
            // If not, Delete the entry so we can regen it as a string
            //
            if (REG_SZ != dwType)
                ::RegDeleteValue ( m_hKey, (LPTSTR)szName );

            //
            // terminate the string...ensure that we dont go past
            // the max lenth of the string!

```

```

        //
        ((LPTSTR)szValue) [ min(dwSize,uiLenWithNull) ] = 0;
    }
#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegGetStr()

// -----
// Method: RegPutStr()
// Purpose: write the information to the registry (write the NULL TOO).
//
BOOL xdRegistry::RegPutStr ( LPCTSTR szName, LPCTSTR szValue )
{
    BOOL bOK = TRUE;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

#ifdef _VXD_SOURCE_
        //
        // move everything into a temp buffer so that we can ensure
        // the existence of a NULL byte on the end of the string
        //
        CString sTmp;
        LPTSTR szBuf = sTmp.GetBuffer(512);
        memset ( szBuf, 0, 512 );
        memcpy ( szBuf, szValue, min(sTmp.GetAllocLength()-1,strlen(szValue)) );

        //
        // remember...always write the NULL byte too!
        //
        UINT uiLenWithNull = strlen(szBuf) + 1;
        m_lRetCode = ::RegSetValueEx ( m_hKey, (LPTSTR)szName, 0, REG_SZ,
                                     (LPBYTE)szBuf,
                                     uiLenWithNull );
    }
    #else
        #ifdef _UNICODE
            char sShort[512];
            char sShortVal[512];
            char sDefault[512];
            BOOL b;
            *sDefault = *sShort=0;
            WideCharToMultiByte ( CP_ACP, 0, szName, -1, sShort, 512, sDefault, &b );
            WideCharToMultiByte ( CP_ACP, 0, szValue, -1, sShortVal, 512, sDefault, &b );
            m_lRetCode = ::RegSetValueExA ( m_hKey, sShort, 0, REG_SZ,

```

```
strlen(sShortVal)+1 );
```

```
(LPBYTE) sShortVal,
```

```
#else
```

```
    CString sTmp;
    LPTSTR szBuf = (LPTSTR)sTmp.GetBuffer(1024);
    memset ( szBuf, 0, 1024 );
    memcpy ( szBuf, szValue, min(1023,_tcslen(szValue))*sizeof(TCHAR) );
    szBuf[_tcslen(szValue)] = 0;
```

```
//
```

```
// remember...always write the NULL byte too!
```

```
//
```

```
    UINT    uiLenWithNull = _tcslen(szBuf) + 1;
```

```
    m_lRetCode = ::RegSetValueEx ( m_hKey, szName, 0, REG_SZ,
```

```
    uiLenWithNull );
```

```
(LPBYTE) szBuf,
```

```
#endif
```

```
#endif
```

```
    bOK = (ERROR_SUCCESS == m_lRetCode);
```

```
#ifndef _VXD_SOURCE_
```

```
{
```

```
    catch(...)
```

```
{
```

```
        XDCATCH;
```

```
        bOK = FALSE;
```

```
}
```

```
#endif
```

```
    return bOK;
```

```
} // End of RegPutStr()
```

```
// -----
```

```
// Method: RegGetNum()
```

```
// Purpose: Retrieves a number from the registry. there are various
```

```
//           overloads for different types.
```

```
//
```

```
BOOL xdRegistry::RegGetNum(LPCTSTR sName, DWORD& dwValue)
```

```
{
```

```
    BOOL bOK = TRUE;
```

```
    CString sTmp;
```

```
    LPTSTR    szBuf = sTmp.GetBuffer(XD_LEN_64);
```

```
    memset ( szBuf, 0, XD_LEN_64 );
```

```
    DWORD dwType = 0;
```

```
    DWORD dwSize = XD_LEN_64-1;
```

```
//
```

```
// Make sure that the registry is open
```

```
//
```

```
    if (m_hKey == NULL)
```

```
        return FALSE;
```

```
#ifndef _VXD_SOURCE_
```

```
    try
```

```
{
```

```
#endif
```

```
#ifdef _VXD_SOURCE_
```

```
    bOK = RegGetStr ( sName, szBuf, sTmp.GetAllocLength()-1 );
```

```
    if ( bOK == TRUE )
```

```

        dwValue = (DWORD)atol((LPTSTR)szBuf);
#else
    #ifdef _UNICODE
        char sShort[512];
        char sDefault[512];
        char bufTmp[512];
        BOOL b=0;
        *sDefault = *sShort=0;
        WideCharToMultiByte ( CP_ACP, 0, sName, -1, sShort, 512, sDefault, &b );
        m_lRetCode = ::RegQueryValueExA (m_hKey,          // hive/key
                                         sShort,
                                         0,
                                         &dwType,
                                         (LPBYTE)bufTmp,
                                         &dwSize);

        // value name

        // reserved

        // the REG_* type

        // pointer to the storage area

        // # to fetch (WITH NULL)
        bOK = (ERROR_SUCCESS == m_lRetCode);
        if ( bOK == TRUE )
        {
            if ( dwType == REG_SZ )
                dwValue = (DWORD)atol(bufTmp);
        }
    #else
        m_lRetCode = ::RegQueryValueEx (    m_hKey,

                                           sName,
                                           0,
                                           &dwType,
                                           (BYTE*)szBuf,
                                           &dwSize );

        bOK = (ERROR_SUCCESS == m_lRetCode);
        if ( bOK == TRUE )
        {
            if ( dwType == REG_SZ )
                dwValue = (DWORD)_ttoi((LPTSTR)szBuf);
            if ( dwType == REG_DWORD )
                dwValue = * ((DWORD*)szBuf);
        }
    #endif
#endif

    #ifndef _VXD_SOURCE_
    {
        XDCATCH;
        bOK = FALSE;
    }
    #endif

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. UINT version
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, UINT& uiValue)
{

```

```

        DWORD      dwValue = uiValue;
        BOOL  bOK = RegGetNum(sName,dwValue);

        uiValue = (UINT) dwValue;

        return bOK;
    } // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. BOOL version
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, BOOL& bValue)
{
    DWORD      dwValue = bValue;
    BOOL  bOK = RegGetNum(sName,dwValue);

    bValue = (BOOL) dwValue;

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. WORD VERSION.
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, WORD& wValue)
{
    DWORD      dwValue = wValue;
    BOOL  bOK = RegGetNum(sName,dwValue);

    wValue = (WORD) dwValue;

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegPutNum()
// Purpose: writes a numeric value to the registry.
//
BOOL xdRegistry::RegPutNum(LPCTSTR sName, DWORD dwValue)
{
    BOOL  bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

#ifdef _VXD_SOURCE_
        CString sTmp;
        BYTE* szBuf = (BYTE*)sTmp.GetBuffer(132);
        sprintf( (LPTSTR)szBuf, _T("%lu"), dwValue);
        UINT uiLenWithNull = strlen((LPTSTR)szBuf) + 1; // ADD THE NULL!!!!!!
        m_lRetCode = ::RegSetValueEx ( m_hKey, (LPTSTR)sName,

```



```

uiLenWithNull );
    bOK = (ERROR_SUCCESS == m_lRetCode);
#else
    #ifdef _UNICODE
        char sShort[512];
        char sDefault[512];
        BOOL b;
        *sDefault = *sShort=0;
        WideCharToMultiByte ( CP_ACP, 0, sName, -1, sShort, 512, sDefault, &b );
        sprintf( sDefault, "%lu", dwValue );
        m_lRetCode = ::RegSetValueExA ( m_hKey, sShort, 0, REG_SZ,
                                                    (LPBYTE)sDefault,
                                                    strlen(sDefault)+1 );
    #else
        CString sTmp;
        LPTSTR szBuf = sTmp.GetBuffer(XD_LEN_64);
        wsprintf( (LPTSTR)szBuf, _T("%lu"), dwValue);
        UINT uiLenWithNull = _tcslen((LPTSTR)szBuf) + 1; // ADD THE NULL!!!!!!
        m_lRetCode = ::RegSetValueEx ( m_hKey,
                                                    sName,
                                                    0,
                                                    REG_SZ,
                                                    (BYTE*)szBuf,
                                                    uiLenWithNull);
    #endif
#endif

    bOK = (ERROR_SUCCESS == m_lRetCode);

#ifdef _VXD_SOURCE_
}
catch(...)
{
    XDCATCH;
    bOK = FALSE;
}
#endif

return bOK;
} // End of RegPutNum()

// -----
// Method: RegDeleteValue()
// Purpose:
//
BOOL xdRegistry::RegDeleteValue ( LPCTSTR szValue )
{
    BOOL bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
try
{
#endif
    m_lRetCode = ::RegDeleteValue ( m_hKey, (LPTSTR)szValue );

```

```

        bOK = (ERROR_SUCCESS == m_lRetCode);

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegDeleteValue()

// -----
// Method: RegEnumVal()
// Purpose: enumerates values for a key. i is the index to get
//
BOOL xdRegistry::RegEnumVal ( int i, LPCTSTR szValueName, UINT uiNameLenWithNull,
                             LPCTSTR szValueData, UINT
uiDataLenWithNull)
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiNameLenWithNull;
    DWORD dwDataSize = (DWORD) uiDataLenWithNull;
    LPBYTE pValue = (LPBYTE) szValueName;
    LPBYTE pDataValue = (LPBYTE) szValueData;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiNameLenWithNull );
        memset ( pDataValue, 0, uiDataLenWithNull );

#ifdef _VXD_SOURCE_
        m_lRetCode = ::RegEnumValue(m_hKey,
                                     // hive/key
                                     dwIdx,
                                     // index
                                     (LPTSTR)pValue,
                                     //
                                     &dwSize,
                                     // the
                                     0,
                                     NULL,
                                     //
                                     pDataValue,
                                     &dwDataSize);

        #else
        m_lRetCode = ::RegEnumValue(m_hKey,
                                     // hive/key

```

```

of the value to get          dwIdx,          // index
valuename will go here      (LPTSTR)pValue,    //
size of the buffer          &dwSize,          // the
                             0,
                             // reserved,
address of type code         NULL,             //
                             pDataValue,
                             &dwDataSize);
#endif

```

```

        bOK = (ERROR_SUCCESS == m_IRetCode);
        if ( bOK == TRUE )
        {
            //
            // terminate the string...ensure that we dont go past
            // the max lenh of the string!
            //
            ((LPTSTR)szValueName) [ min(dwSize,uiNameLenWithNull) ] = 0;
            ((LPTSTR)szValueData) [ min(dwDataSize,uiDataLenWithNull) ] = 0;
        }

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegEnumVal()

```

```

// -----
// Method: RegPutBin()
// Purpose: write the information to the registry
//
BOOL xdRegistry::RegPutBin ( LPCTSTR szName, BYTE* pBuffer, UINT uiLength )
{

```

```

    BOOL bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

```

```

#ifdef _VXD_SOURCE_
    try
    {

```

```

#endif

```

```

        //
        // move everything into a temp buffer so that we can ensure
        // the existance of a NULL byte on the end of the string
        //
        CString sTmp;
        LPTSTR szBuf = sTmp.GetBuffer(132);
        memset ( szBuf, 0, 132 );

```

```

        memcpy ( szBuf, pBuffer, min(sTmp.GetAllocLength()-1,uiLength) );

        m_lRetCode = ::RegSetValueEx ( m_hKey,

                                     (LPTSTR)szName,
                                     0,
                                     REG_BINARY,
                                     (LPBYTE) szBuf,
                                     uiLength );

        bOK = (ERROR_SUCCESS == m_lRetCode);

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegPutBin()

// -----
// Method: RegEnumKey()
// Purpose: enumerates values for a key. i is the index to get
//
BOOL xdRegistry::RegEnumKey ( int i, LPCTSTR szValueName, UINT uiNameLenWithNull)
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiNameLenWithNull;
    LPBYTE pValue = (LPBYTE) szValueName;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiNameLenWithNull );

#ifdef _VXD_SOURCE_
        m_lRetCode = ::RegEnumKey(m_hKey,

                                // hive/key
                                dwIdx,                // index of the
                                (LPTSTR)pValue,        // valuenam will
                                dwSize);               // the size of the

        // hive/key
        dwIdx,                // index of the
        (LPTSTR)pValue,        // valuenam will

```

```
buffer                                     dwSize);                               // the size of the
#endif

    bOK = (ERROR_SUCCESS == m_IRetCode);
    if (bOK==TRUE)
    {
        //
        // terminate the string...ensure that we dont go past
        // the max lenth of the string!
        //
        ((LPTSTR)szValueName) [ min(dwSize,uiNameLenWithNull) ] = 0;
    }

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegEnumKey()
```

//

// **Module: xdFileIO.cpp**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Redefinitions for the FILE IO functions

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnowWare(r), Inc.

// All rights reserved.

// -----

// -----

//

#include "stdafx.h"

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h>

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

#ifdef _VXD_SOURCE_

#include LOCKED_CODE_SEGMENT

#include LOCKED_DATA_SEGMENT

#endif

#ifdef _VXD_SOURCE_

// -----

// Function: CreateFile()

// Purpose: This API function maps the standard Win32 CreateFile function

// to the Ring-0 R0_OpenCreateFile() call.

// Returns: INVALID_HANDLE_VALUE - bad

// something else - good!

//

HANDLE CreateFile (LPCTSTR lpFileName, // pointer to name of the file
 DWORD dwDesiredAccess, // access (read-write) mode
 DWORD dwShareMode, // share mode
 void* lpSecAtt, // pointer to security

attributes

DWORD dwCreateFlags, // how to create
 DWORD dwFlagsAndAttributes, // file attributes
 HANDLE)

{

HANDLE h = INVALID_HANDLE_VALUE;

WORD wError = 0;

WORD wMode = 0;

BYTE action = 0;

switch (dwDesiredAccess)

{

case GENERIC_READ:

wMode = OPEN_ACCESS_READONLY;

break;

case GENERIC_WRITE:

wMode = OPEN_ACCESS_WRITEONLY;

break;

default:

wMode = OPEN_ACCESS_READWRITE;

break;

}

```

//
// file sharing not supported!
//
wMode |= OPEN_SHARE_COMPATIBLE;

//
// Create Attributes
//
switch ( dwCreateFlags )
{
case CREATE_NEW: // create New file. fail if file exists
    action = ACTION_IFEXISTS_FAIL | ACTION_IFNOTEXISTS_CREATE;
    break;
case CREATE_ALWAYS: // create New file. overwrite if exists
    action = ACTION_IFEXISTS_TRUNCATE | ACTION_IFNOTEXISTS_CREATE;
    break;
case OPEN_EXISTING: // open file, fail if the file does not exists
    action = ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_FAIL;
    break;
case OPEN_ALWAYS: // open file. if !exists, create
    action = ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_CREATE;
    break;
case TRUNCATE_EXISTING: // open&truncate file. fail if it does not exist
    action = ACTION_IFEXISTS_OPEN | ACTION_IFEXISTS_TRUNCATE |
ACTION_IFNOTEXISTS_FAIL;
    break;
}

h = R0_OpenCreateFile(1,(LPTSTR)lpFileName,wMode,
                     ATTR_NORMAL,action,R0_NO_CACHE,&wError,
&action);
    return h;
} // End of CreateFile()

// -----
// Function: ReadFile()
// Purpose: This API function maps the standard Win32 ReadFile function
//          to the Ring-0 R0_ReadFile() call.
// Returns: TRUE - Good read
//          FALSE - Bad Read
//
BOOL ReadFile ( HANDLE hFile, void* lpBuffer, DWORD dwBytesToRead,
                DWORD* pdwBytesRead, void* pdwOffset)
{
    WORD wError = 0;
    DWORD dwOffset = 0;

    if ( pdwOffset )
        dwOffset = *((DWORD*)pdwOffset);

    *pdwBytesRead = R0_ReadFile ( TRUE, hFile, lpBuffer, dwBytesToRead,
                                dwOffset, &wError );

    return ( wError == 0 );
} // End of ReadFile()

// -----
// Function: WriteFile()
// Purpose: This API function maps the standard Win32 WriteFile function
//          to the Ring-0 R0_WriteFile() call.
// Returns: TRUE - Good write
//          FALSE - Bad write

```

```

//
BOOL WriteFile ( HANDLE hFile, LPCTSTR lpBuffer, DWORD dwBytesToWrite,
                DWORD* pBytesWritten, void* p)
{
    WORD wError = 0;
    DWORD dwFilePos = R0_GetFileSize(hFile,&wError);
    *pBytesWritten = R0_WriteFile ( TRUE, hFile, (void*)lpBuffer, dwBytesToWrite,
                                   dwFilePos, &wError );
    return (wError == 0);
} // End of WriteFile()

// -----
// Function: CloseHandle()
// Purpose: This API function maps the standard Win32 CloseHandle function
//          to the Ring-0 R0_CloseFile() call.
// Returns: TRUE - success
//          FALSE - failure
//
BOOL CloseHandle ( HANDLE hFile )
{
    WORD wError = 0;
    return R0_CloseFile ( hFile, &wError );
} // End of CloseHandle()

// -----
// Function: GetFileSize()
// Purpose: This API function maps the standard Win32 GetFileSize function
//          to the Ring-0 R0_GetFileSize() call.
// Returns: TRUE - success
//          FALSE - failure
//
DWORD GetFileSize ( HANDLE hFile, DWORD* pdwHigh )
{
    WORD wError = 0;
    return R0_GetFileSize ( hFile, &wError );
} // End of GetFileSize()

// -----
// Function: ReadFileLine()
// Purpose: This API function maps the standard Win32 ReadFile function
//          to the Ring-0 R0_ReadFile() call.
// Returns: TRUE - Good read
//          FALSE - Bad Read
//
BOOL ReadFileLine ( HANDLE hFile, BYTE* lpBuffer,
                   DWORD dwBytesToRead,
                   DWORD* pdwBytesRead,
                   DWORD* pdwOffset )
{
    WORD wError = 0;
    DWORD dwOffset = 0;

    if ( pdwOffset )
        dwOffset = *((DWORD*)pdwOffset);

    //
    // Check for EOF
    //
    if ( dwOffset >= R0_GetFileSize(hFile,&wError) )
        return FALSE;
}

```



```
// *pdwBytesRead = R0_ReadFile ( TRUE, hFile, lpBuffer, dwBytesToRead,
//                               dwOffset, &wError );

memset ( lpBuffer, 0, dwBytesToRead );

int iTmpBytesRead = 1;
BOOL bFoundEOL = FALSE;
int i=0;
for ( i=0; (iTmpBytesRead != 0) && (i<dwBytesToRead) &&
      (wError == 0) && (bFoundEOL==FALSE); i++ )
{
    iTmpBytesRead = R0_ReadFile ( TRUE, hFile, &(lpBuffer[i]), 1, dwOffset+i, &wError );
    if ((iTmpBytesRead != 0) && (wError == 0))
    {
        if ( lpBuffer[i] == chNL )
            bFoundEOL = TRUE;
    }
}

*pdwBytesRead = i;

return ( wError == 0 );
} // End of ReadFileLine()
#endif
```

//

// **Module: xdDebugger.cpp**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Implementation module for the xdDebugger utility class.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnowWare(r), Inc.

// All rights reserved.

//

// -----

//

#include "stdafx.h"

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h> // X:drive Tools Related

#include <xdDebugger.h>

#ifndef _VXD_SOURCE_

#include <afxmt.h>

#include "resource.h"

#endif

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

#ifdef _VXD_SOURCE_

#include LOCKED_CODE_SEGMENT

#include LOCKED_DATA_SEGMENT

#endif

// -----

// Method: xdDebugger()

// Purpose: Constructor for the debugger class.

//

xdDebugger::xdDebugger()

{

#ifndef _VXD_SOURCE_

try

#endif

{

m_szLogFile = (LPTSTR)malloc(XD_LEN_1024);

m_szMsg = (LPTSTR)malloc(XD_LEN_2048);

m_szBuf = (LPTSTR)malloc(XD_LEN_2048);

m_hLogFile = NULL;

m_bLogFile = FALSE;

_tcscpy (m_szLogFile, XD_LOGFILE_NP);

#ifndef _VXD_SOURCE_

m_pSem = new CSemaphore(1,1);

#endif

}

#ifndef _VXD_SOURCE_

catch(...)

{

XDCATCH;

```

    }
#endif
} // End of xdDebugger()

// -----
// Method: ~xdDebugger()
// Purpose: Destructor.
//
xdDebugger::~xdDebugger()
{
#ifdef _VXD_SOURCE_
    delete m_pSem;
#endif
    free(m_szMsg);
    free(m_szLogFile);
    free(m_szBuf);
} // End of ~xdDebugger()

// -----
// Method: DebuggerOn()
// Purpose: turns on debugging to the optional logfile
//
void xdDebugger::DebuggerOn(BOOL bInitialize)
{
#ifdef _VXD_SOURCE_
    WORD wError = 0;
    BYTE bAction = 0;

    //
    // force a creation of the file if it does not already exist. Then
    // simply close the file; we'll open it when we need to write to
    // it.
    //
    m_bLogFile = TRUE;
    if ( bInitialize == TRUE )
    {
        LPTSTR szOldFile = (LPTSTR)malloc( XD_LEN_1024 );
        strcpy ( szOldFile, m_szLogFile );
        LPTSTR pDot = strchr(szOldFile, chPERIOD);
        if ( pDot != NULL )
            *pDot = NULL;
        strcat ( szOldFile, ".old" );
        R0_DeleteFile ( szOldFile, 0, &wError );
        R0_RenameFile ( m_szLogFile, szOldFile, &wError );
        m_hLogFile = R0_OpenCreateFile ( TRUE, m_szLogFile,
OPEN_SHARE_DENYWRITE|OPEN_ACCESS_WRITEONLY,
ATTR_NORMAL,
ACTION_IFEXISTS_TRUNCATE|
ACTION_IFNOTEXISTS_CREATE,
0, &wError,
(PUCHAR)&bAction );
        free(szOldFile);
    }
    else
        m_hLogFile = R0_OpenCreateFile ( TRUE, m_szLogFile,
OPEN_SHARE_DENYWRITE|OPEN_ACCESS_WRITEONLY,
ATTR_NORMAL,
```

```

ACTION_IFEXISTS_OPEN|
ACTION_IFNOTEXISTS_CREATE,
(PUCHAR)&bAction );
    //
    // Ok, we opened/created the close it again. We never want to keep
    // the logfile open so that we ensure that its contents are saved
    // to disk.
    //
    if ( (m_hLogFile != NULL) && (wError == 0) )
        R0_CloseFile ( m_hLogFile, &wError );
    m_hLogFile = NULL;
#else
    try
    {
        //
        // force a creation of the file if it does not already exist. Then
        // simply close the file; we ll open it when we need to write to
        // it.
        //
        m_bLogFile = TRUE;

        if (blInitialize == TRUE)
        {
            CString sOldFile;
            LPTSTR szOldFile = sOldFile.GetBuffer(512);
            _tcscpy ( szOldFile, m_szLogFile );
            LPTSTR pDot = _tcsrchr(szOldFile,chPERIOD);
            if ( pDot != NULL)
                *pDot = NULL;
            _tcscat ( szOldFile, _T(".old") );
            DeleteFile ( szOldFile );
            try
            {
                CFile::Rename( m_szLogFile, szOldFile );
            }
            catch(...)
            {
            }
        }
#ifdef _UNICODE
        m_hLogFile = _wfopen(m_szLogFile,_T("w+"));
#else
        m_hLogFile = fopen(m_szLogFile,_T("w+"));
#endif
    }
    else
#ifdef _UNICODE
        m_hLogFile = _wfopen(m_szLogFile,_T("a+"));
#else
        m_hLogFile = fopen(m_szLogFile,_T("a+"));
#endif
    if ( m_hLogFile != NULL )
        fclose(m_hLogFile);
    m_hLogFile = NULL;
}
catch(...)
{
    XDCATCH;
}
#endif

```

```

} // End of DebuggerOn()

// -----
// Method: DebuggerOff()
// Purpose: turns off debugging to the optional logfile
//
void xdDebugger::DebuggerOff()
{
#ifdef _VXD_SOURCE
    WORD wError = 0;
    if (m_hLogFile!=NULL)
        R0_CloseFile ( m_hLogFile, &wError );
    m_bLogFile = FALSE;
#else
    m_bLogFile = FALSE;
#endif
} // End of DebuggerOff()

// -----
// Method: DEBUGMSG()
// Purpose: always dumps the messages to debugger window and optionally to
//         the file...
//
void xdDebugger::DEBUGMSG(TCHAR *fmt,...)
{
#ifdef _VXD_SOURCE_
    va_list      args;
    //
    // parse out the info
    //
    va_start(args,fmt);
    vsprintf(m_szBuf,fmt,args);
    va_end(args);
    //
    // add a <cr>
    //
    if (strchr(m_szBuf,chNL)==NULL)
        strcat(m_szBuf,"r\n");

    strcpy ( m_szMsg, "FSD: ");
    strcat ( m_szMsg, m_szBuf );

#ifdef DEBUG
    DEBUGTRACE(m_szMsg);
#endif

    //
    // if the logfile is engaged, dump it!
    //
    if (m_bLogFile==TRUE)
    {
        WORD wError = 0;
        BYTE bAction = 0;

        //
        // open the file, dump the string, then close the file!!!
        //
        m_hLogFile = R0_OpenCreateFile ( TRUE, m_szLogFile,

```

```

OPEN_SHARE_DENYWRITE|OPEN_ACCESS_WRITEONLY,

ATTR_NORMAL,

ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_CREATE,

(PUCHAR)&bAction);
    if ((m_hLogFile != NULL) && (wError == 0))
    {
        DWORD dwOffset = R0_GetFileSize ( m_hLogFile, &wError );
        R0_WriteFile ( TRUE, m_hLogFile, m_szMsg, strlen(m_szMsg),
                                dwOffset, &wError);
        R0_CloseFile(m_hLogFile,&wError);
        m_hLogFile = NULL;
    }
}
#else
try
{
    //
    // only wait 1 second, then do it. This guarantees that
    // we dont lock up the system
    //
    if ( m_pSem->Lock(5000) == TRUE )
    {
        va_list      args;
        //
        // parse out the info
        //
        va_start(args,fmt);
        _vstprintf(m_szBuf,fmt,args);
        va_end(args);
        //
        // add a <cr>
        //
        if ( _tcschr(m_szBuf,chNL)==NULL)
            _tcscat(m_szBuf,szNL);

        *m_szMsg = 0;
        _tcscpy(m_szMsg,_T("LOG: "));
        _tcscat(m_szMsg,m_szBuf);

        //
        // dump it to the IDE debugger
        //
        #ifdef _DEBUG
            OutputDebugString(m_szMsg);
        #endif

        //
        // if the logfile is engaged, dump it!
        //
        if (m_bLogFile == TRUE)
        {
            //
            // open the file, dump the string, then close the file!!!
            //

            #ifdef _UNICODE
                m_hLogFile = _wfopen(m_szLogFile,_T("a"));
            #else
                m_hLogFile = fopen(m_szLogFile,_T("a"));
            #endif
        }
    }
}
catch (...)
{
}

```

```

#endif
        if (m_hLogFile != NULL)
        {
            _fputs(m_szMsg,m_hLogFile);
            //fflush(m_hLogFile);
            fclose(m_hLogFile);
            m_hLogFile = NULL;
        }
    }
}
catch(...)
{
    XDCATCH;
    if (m_hLogFile!=NULL)
    {
        //fflush(m_hLogFile);
        fclose(m_hLogFile);
        m_hLogFile = NULL;
    }
}
m_pSem->Unlock();
#endif

} // End of DEBUGMSG

#ifdef _VXD_SOURCE_
// -----
// Method: DEBUGMSG()
// Purpose: loads the string and then dumps it to the logfile.
//
void xdDebugger::DEBUGMSG(UINT uiResourceId)
{
    CString s = XD_LOADSTRING(uiResourceId);
    DEBUGMSG( T("%s\n"),s);
} // End of DEBUGMSG()
#endif

// -----
// Method: SetLogName()
// Purpose:
//
void xdDebugger::SetLogName(LPCTSTR s)
{
    _tcscpy ( m_szLogFile, s );
} // End of SetLogName()

// -----
// Method: IsDebuggerOn()
// Purpose:
//
BOOL xdDebugger::IsDebuggerOn ( void )
{
    return m_bLogFile;
} // End of IsDebuggerOn()

```

JavaScript Listing

//button.js	1
//diskInfo.js	6
//launch.js.....	10
//nav.js.....	11
//saveToXdrive.js.....	28
//secure_login.js	31
//skip.js.....	33
//skipthownload.js	35
//submit.js.....	38
//uploadStatus.js.....	53
//utils.js.....	54
//verify_lib.js.....	57
//xparse.js	69

//button.js

```

// Is called upon loading of page to set up the button image arrays
function XDloadToolbarButtons ()
{
    if (XD_gsAction == '') {
        for (var i=0; i < 4; i=i+3)
        {
            g_aimgUpload[i] = new Image();
            g_aimgDownload[i] = new Image();
            g_aimgNewFolder[i] = new Image();
            g_aimgMove[i] = new Image();
            g_aimgRename[i] = new Image();
            g_aimgDelete[i] = new Image();
            g_aimgHelp[i] = new Image();
            g_aimgView[i] = new Image();
            g_aimgShare[i] = new Image();

            g_aimgUpload[i].src = XD_gsGraphicsLanguageRoot+ "up" + i +
".gif";
            g_aimgDownload[i].src = XD_gsGraphicsLanguageRoot+ "down" +
i + ".gif";
            g_aimgView[i].src = XD_gsGraphicsLanguageRoot+ "view" + i +
".gif";
            g_aimgNewFolder[i].src = XD_gsGraphicsLanguageRoot+ "new" +
i + ".gif";
            g_aimgMove[i].src = XD_gsGraphicsLanguageRoot+ "move" + i +
".gif";
            g_aimgRename[i].src = XD_gsGraphicsLanguageRoot+ "name" + i
+ ".gif";
            g_aimgDelete[i].src = XD_gsGraphicsLanguageRoot+ "delete" +
i + ".gif";
            g_aimgShare[i].src = XD_gsGraphicsLanguageRoot+ "share" +
i + ".gif";

            // g_aimgUpload[i].src = XD_gsGraphicsLanguageRoot+
"nav_upload" + i + ".gif";
            // g_aimgDownload[i].src = XD_gsGraphicsLanguageRoot+
"nav_download" + i + ".gif";
            // g_aimgView[i].src = XD_gsGraphicsLanguageRoot+ "nav_view" +
i + ".gif";
            // g_aimgNewFolder[i].src = XD_gsGraphicsLanguageRoot+
"nav_newfolder" + i + ".gif";
            // g_aimgMove[i].src = XD_gsGraphicsLanguageRoot+ "nav_move" +
i + ".gif";
            // g_aimgRename[i].src = XD_gsGraphicsLanguageRoot+
"nav_rename" + i + ".gif";
            // g_aimgDelete[i].src = XD_gsGraphicsLanguageRoot+
"nav_delete" + i + ".gif";
            // g_aimgShare[i].src = XD_gsGraphicsLanguageRoot+
"nav_share" + i + ".gif";
        }
    }
}

// Takes a button and an event and returns a status
// as defined by the containt button statuses
function XDtoolbarButtonStatus(button, event)
{
    var rv = XD_TOOLBAR_BUTTON_ENABLED;

```

```

// Just exit if no controls are enabled
if(!ControlsEnabled)
{
    return XD_TOOLBAR_BUTTON_DISABLED;
}

if (event == XD_EVENT_MOUSEOVER)
{
    rv = XD_TOOLBAR_BUTTON_ACTIVE;
}
else if (event == XD_EVENT_MOUSEOUT)
{
    rv = XD_TOOLBAR_BUTTON_ENABLED;
}
else if (event == XD_EVENT_CLICK)
{
    rv = XD_TOOLBAR_BUTTON_CLICKED;
}

if ((button == XD_TOOLBAR_BUTTON_UPLOAD)
&& (XD_gnSelectedFolderCount != 1))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    ((button == XD_TOOLBAR_BUTTON_DOWNLOAD)
    && (XD_gnSelectedCount != 1 || XD_gnSelectedFolderCount != 0))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if

    ((button == XD_TOOLBAR_BUTTON_NEWFOLDER)
    && (XD_gnSelectedFolderCount != 1))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if

    ((button == XD_TOOLBAR_BUTTON_MOVE)
    && (XD_gnSelectedCount == 0))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if

    ((button == XD_TOOLBAR_BUTTON_DELETE)
    && (XD_gnSelectedCount == 0))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    ((button == XD_TOOLBAR_BUTTON_RENAME)
    && (XD_gnSelectedCount != 1))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    (button == XD_TOOLBAR_BUTTON_VIEW)
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
    if (XD_gnSelectedCount == 1 && XD_gnSelectedFolderCount == 0)
    {

```

```

        rv = XD_TOOLBAR_BUTTON_ENABLED;
    }
}
else if
    (button == XD_TOOLBAR_BUTTON_SHARE)
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
    if (XD_gnSelectedCount == 1 && XD_gnSelectedFolderCount == 0)
    {
        rv = XD_TOOLBAR_BUTTON_ENABLED;
    }
}

return rv;
}

// Wrapper for updating images, used for checking if the image exists before
// attempting to update it.

function XDImageUpdate (oImage,imgGraphic)
{
    if (oImage)
    {
        // If the image exists then update it
        oImage.src = imgGraphic;
    }
    else
    {
        // otherwise do nothing
    }
}

// Takes a button and an event, finds the status
// and then refreshes the button.

function XDrefreshButton (sButton, sEvent)
{
    if (XD_gsAction == '') {
        var nStatus = XDtoolbarButtonStatus(sButton, sEvent);
        var oFrame = XD_goFrameControls;

        XD_gsPreviousGrove = grove;

        if (sButton == XD_TOOLBAR_BUTTON_UPLOAD)
        {
            XDImageUpdate(oFrame.document.img_upload,g_aimgUpload[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_DOWNLOAD)
        {
            XDImageUpdate(oFrame.document.img_download,g_aimgDownload[nStatus].src)
;
        }
        else if (sButton == XD_TOOLBAR_BUTTON_NEWFOLDER)
        {
            XDImageUpdate (oFrame.document.img_newfolder,
g_aimgNewFolder[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_MOVE)
        {
            XDImageUpdate(oFrame.document.img_move,g_aimgMove[nStatus].src);
        }
    }
}

```

```

    else if (sButton == XD_TOOLBAR_BUTTON_RENAME)
    {

        XDImageUpdate(oFrame.document.img_rename,g_aimgRename[nStatus].src);
    }
    else if (sButton == XD_TOOLBAR_BUTTON_DELETE)
    {

        XDImageUpdate(oFrame.document.img_delete,g_aimgDelete[nStatus].src);
    }
    else if (sButton == XD_TOOLBAR_BUTTON_VIEW)
    {
        XDImageUpdate(oFrame.document.img_view,g_aimgView[nStatus].src);
    }
    else if (sButton == XD_TOOLBAR_BUTTON_SHARE)
    {

        XDImageUpdate(oFrame.document.img_share,g_aimgShare[nStatus].src);
    }
}

//This refreshes all the buttons at one time.
function XDrefreshAllButtons()
{
    XDrefreshButton(XD_TOOLBAR_BUTTON_UPLOAD, null);
    XDrefreshButton(XD_TOOLBAR_BUTTON_DOWNLOAD, null);
    XDrefreshButton(XD_TOOLBAR_BUTTON_NEWFOLDER, null);
    XDrefreshButton(XD_TOOLBAR_BUTTON_MOVE, null);
    XDrefreshButton(XD_TOOLBAR_BUTTON_RENAME, null);
    XDrefreshButton(XD_TOOLBAR_BUTTON_DELETE, null);
    XDrefreshButton(XD_TOOLBAR_BUTTON_VIEW, null);
    XDrefreshButton(XD_TOOLBAR_BUTTON_SHARE, null);
}

// Wrapper that handles button click events.
function XDbuttonClick (sButton)
{
    XDrefreshButton(sButton, XD_EVENT_CLICK);
}

// Wrapper that handles the button MouseOver events
function XDbuttonOver (sButton)
{
    XDrefreshButton(sButton, XD_EVENT_MOUSEOVER);
}

// Wrapper that handles teh button MouseOut events.
function XDbuttonOut (sButton)
{
    XDrefreshButton(sButton, XD_EVENT_MOUSEOUT);
}

function XDfunctionStatus(button)
{
    if (! ControlsEnabled)
    {
        return false;
    }
}

```

```
    if (XDtoolbarButtonStatus(button, XD_EVENT_MOUSEOVER) ==  
XD_TOOLBAR_BUTTON_ACTIVE)  
    {  
        return true;  
    }  
    else  
    {  
        return false;  
    }  
}
```

//diskInfo.js

```
// NOTE: The table trick works differently in IE vrs Netscape. In netscape
you need to
// have an &nbsp; as a value within the TD's while in IE you do not need
anything.
function mresponse()
{
    parent.parent.parent.frames['centerview'].document.location =
    "../explorer/more_space_mail.html";
}

function XDdisplayDiskInfo (oFrame)
{
    //3K always taken up by xdrive, public and private folders
    //changed code so it doesn't show as red any more
    var nUsed = XD_gnQuotaUsed;
    var nTotal = XD_gnQuotaTotal;

    //var nGraphWidth = XD_gnFileGraphWidth;
    var sGraphUsedColor = XD_gsUsedColor;
    var sGraphFreeColor = XD_gsFreeColor;

    var freeMB = nTotal - nUsed;
    var usedPercent = Math.round(100 * (nUsed/nTotal));

    ///// Do some basic bound checking
    if (usedPercent > 100)
    {
        usedPercent = 100;
        sGraphFreeColor = sGraphUsedColor;
    }
    if ( usedPercent < 0 )
    {
        usedPercent = 0;
    }

    var freePercent = 100-usedPercent;

    oFrame.write('<FORM name="controlForm">');

    oFrame.write('<TABLE width=500 border=0 cellpadding=0
cellspacing=0><TR>\n');
    oFrame.write('<TD width=300>&nbsp;</TD>\n');
    oFrame.write('<TD align="right" width=50><B><FONT size="-1">' +
XD_gsEmpty + '</FONT></B></TD>\n');
    oFrame.write('<TD align="center" width=100>\n');
    oFrame.write('<TABLE width=100 CELLPADDING=0 CELLSPACING=0
BORDER=0><TR>\n');
    if (usedPercent != 0)
    {
        oFrame.write('<TD height=10 WIDTH="" + usedPercent + '%' BGCOLOR="" +
sGraphUsedColor + '></TD>\n');
    }
    oFrame.write('<TD height=10 WIDTH="" + freePercent + '%' BGCOLOR="" +
sGraphFreeColor + '></TD>\n');
    oFrame.write('</TR></TABLE>\n');
```

```

oFrame.write('</TD><TD align="left" width=50><B><FONT size="-1">' + XD_gsFull +
'</FONT></B></TD>\n');

oFrame.write('</TR>\n');
oFrame.write('</TABLE>\n');

if (usedPercent>90)
{
oFrame.write('<TABLE width=500 border=0 cellpadding=0
cellspace=0><TR><TD width=300></TD><TD width=200 valign=center align=left><FONT size="-1"
face="verdana,arial">' + XD_gsOutOfSpace + '?<BR><A HREF="/cgi-
bin/addspace.cgi?action=intro" target="centerview">' + XD_gsBuyMore +
'</A></FONT></TD></TR></TABLE>');
}

oFrame.write('<input type="hidden" name="multipleSelect" value="N">');
oFrame.write('</FORM>');
}

function XDSelectedList()
{
return XD_gsSelectedList;
}

function XDSelectedFolder()
{
return
XD_gsSelectedFolderList.substring(0,XD_gsSelectedFolderList.length-1);
}

/*****
* XDCleanupPath: Cleanup the passed path by removing the "/X:drive/" prefix
* and the + postfix.
*****/
function XDPathCleanup(sPath)
{
var sCopy = sPath;
sCopy = sCopy.substring(9,sCopy.length)
//sCopy = sCopy.substring(0,sCopy.length-1);
return sCopy;
}

function XDMultiSelect (sValue)
{
if (sValue != 'null' && sValue != "")
{
m_sMultiSelect = sValue;
}
else
{
return m_sMultiSelect;
}
}

function HTMLNavigation ()
{
var sHTML = HTMLStart()
+ '<table width="100%" border="0" cellspacing="0"
cellpadding="0">'

```

```

        +'<tr align="left" valign="top" bgcolor="#5EB114">'
        +'<td></td>'
        +'</tr><tr>'
        +'<td></td>'
        +'</tr><tr>'
        +'<td></td>'
        +'</tr></table>'
        +'<a target="toolbar" href="http://www.mit.edu">MIT</a>'
        +'</BODY>\n</HTML>';
    return sHTML;
}

function HTMLStart ()
{
    return "<HTML>\n"
    +'<body bgcolor="#6961AB" topmargin="0" leftmargin="0" marginheight="0"
marginwidth="0" text="#FFFFFF" vlink="#FFFFFF" alink="#FFFFFF" link="#FFFFFF"
{onload}>'
    +" \n";
}

function HTMLEnd ()
{
    return "\n</BODY>\n</HTML>\n";
}

function RedrawToolBar()
{
    var sWindow = 'window.toolbar';
    sWindow.document.write(HTMLStart()+'test'+HTMLEnd());
}

function roundOff(value, precision)
{
    value += .000000001;
    part = "" + parseInt(value);
    size = part.length;
    value = "" + value; //convert value to string
    return value.substring(0,size+1+precision);
}

function XDDiskUsed()
{
    var nUsed = XD_gnQuotaUsed;
    var nUsedMB = nUsed/1024;
    var nRound = roundOff(nUsedMB,2);
    var sRounded;

    // if (nUsed < 1024)
    // {
    //     sRounded = '.'+nRound;
    // }
    // else
    // {
    //     sRounded = nRound;
    // }

    return sRounded;
}

```



```
function XDDiskTotal()
{
    var nTotal = XD_gnQuotaTotal;
    var nTotalMB = nTotal/1024;
    var nRound = roundOff(nTotalMB,2);
    var sRounded;

    // if (nTotal < 1024)
    // {
    //     sRounded = '.'+nRound;
    // }
    // else
    // {
    //     sRounded = nRound;
    // }

    return sRounded + ' MB';
}

function XDDiskFree()
{
    var nUsed = XD_gnQuotaUsed;
    var nTotal = XD_gnQuotaTotal;

    var nFreeMB = (nTotal - nUsed)/1024;
    var nRound = roundOff(nFreeMB,2);
    var sRounded;

    // if (nFreeMB < 1)
    // {
    //     sRounded = '.'+nRound;
    // }
    // else
    // {
    //     sRounded = nRound;
    // }

    return sRounded + ' MB';
}
```

//launch.js

```

/*****
 * XDEplorerLaunch: Launch the passed explorer URL in a popup window.
 *****/

function XDEplorerLaunch (
    sURL, /*** (I) The URL to open in the popup window
    nHeight, /*** (I) The height of the popup
    nWidth) /*** (I) The width of the popup
    {
        var w =
        window.open(sURL,"XDriveExplorer","location=no,toolbar=no,menubar=yes,"+
            "status=no,resizable=no,scrolling=yes,scrollbars=no,"+
            "width="+nWidth+",height="+nHeight);

        /*** make sure the opener knows who the parent is
        if (w.opener == null) w.opener = self;

        /*** focus on the newly created window
        w.focus();
    }

function XDEplorerURL()
{
    return '/cgi-bin/explorer.cgi';
}

function XDDataURL()
{
    return '/cgi-bin/explorer_data.cgi';
}
```

//nav.js

```
// Added by Julie Wang 111999
//
// Function is used with <a href> to pop up another window to show X:drive's
Terms of Service
// page

function toc()
{
var url, window_name;

url="/company/toc.html";
window_name="toc";
window.open(
    url,
    window_name,

'toolbar=no,menubar=no,scrollbars=yes,fullscreen=no,resizable=no,width=650,height=400'
);
return;
}

// Added by Julie Wang 122199
//
// Function is used with <a href> to pop up another window to show a
// sample letter when someone use "Tell A Friend" feature.

function tell_a_friend_sample_email()
{
var url, window_name;

url="/generic_join_sample_email.html";
window_name="toc";
window.open(
    url,
    window_name,

'toolbar=no,menubar=no,scrollbars=yes,fullscreen=no,resizable=no,width=650,height=400'
);
return;
}

// Added by Julie Wang 102699
//
// Function writes the side bar nav. menu/buttons on general HTML pages for
every visitors.

function left_menu()
{
    document.write('<table width=\\"138\\" border=\\"0\\" cellpadding=\\"0\\" cellspacing=\\"0\\">\n');
    document.write('<tr align=\\"left\\" valign=\\"top\\">\n');
    document.write('<td><a href=\\"http://www.xdrive.com\\"><img src=\\"/graphics/internal/btn_xdrivehome.gif\\" width=\\"138\\" height=\\"19\\" alt=\\"X:drive Home\\" border=\\"0\\"></a><br><img src=\\"/graphics/internal/divider.gif\\" width=\\"138\\" height=\\"5\\" alt=\\"Divider\\"></td>\n');
    document.write('</tr>\n');
```

```

document.write('<tr align="left" valign="top">\n');
document.write('<td>\n');
document.write('<table width="138" border="0" cellspacing="0"
cellpadding="0" background=""/graphics/internal/lines.gif">\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38" height="38"><img src=""/graphics/internal/icon-
new.gif" width="38" height="31" alt="New"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font
face="Arial, Helvetica, sans-serif" size="2"><a
href="/company/new.html">What\'s New</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38" height="34" alt="About X:drive"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font
face="Arial, Helvetica, sans-serif" size="2"><a
href="/company/company.html">About X:drive</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38" height="28"
src=""/graphics/internal/icon_desktop.gif" width="38" height="28"
alt="Desktop X:drive"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font
face="Arial, Helvetica, sans-serif" size="2"><a
href="/company/main_download.html">Desktop X:drive</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38" height="28" alt="Affiliates"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font
face="Arial, Helvetica, sans-serif" size="2"><a
href="/affiliates/befree/index.html">X:drive
Affiliate</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38" height="32" alt="FAQs"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font
face="Arial, Helvetica, sans-serif" size="2"><a
href="/company/faq.html">FAQ</a></font></b></td>\n');
document.write('</tr>\n');
document.write('</table>\n');

document.write('</td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td><a href="/cgi-bin/signup_form.cgi"><img
src=""/graphics/internal/sign-up-now.gif" width="138" height="58"
alt="Sign Up Now !" border="0"></a></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td><img src=""/graphics/internal/whats-hot.gif"
width="138" height="19" alt="What\'s Hot ?"><br><img
src=""/graphics/internal/divider.gif" width="138" height="5"
alt="Divider"></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td><a href="/freebies/english/freebiesout.html"><img
src=""/graphics/internal/freebies.gif" width="138" height="82"
alt="Check Out Freebies - Click Here" border="0"></a><br><img
src=""/graphics/internal/divider.gif" width="138" height="5"
alt="Divider"></td>\n');
document.write('</tr>\n');

```

```

document.write('<tr align=\"left\" valign=\"top\">\n');
document.write('<td><a href=\"/company/main_download.html\"><img
src=\"/graphics/internal/btn_get_application.gif\" width=\"138\"
height=\"82\" alt=\"Download the desktop application!\"
border=\"0\"></a><br><img src=\"/graphics/internal/divider.gif\"
width=\"138\" height=\"5\" alt=\"Divider\"></td>\n');
document.write('</tr>\n');
document.write('<tr align=\"left\" valign=\"top\">\n');
document.write('<td><a href=\"/demo/index.html\"><img
src=\"/graphics/internal/btn_skipdownload.gif\" width=\"138\" height=\"82\"
alt=\"Skip the download!\" border=\"0\"></a><br><img
src=\"/graphics/internal/divider.gif\" width=\"138\" height=\"5\"
alt=\"Divider\"></td>\n');
document.write('</tr>\n');
document.write('</table>\n');

document.close();
return true;
}

```

// Added by Martin Hald

```

function PathRemovePrefix(path)
{
    return path.substring(10,path.length);
}

```

// Function that redraws the file explorer

```

function show()
{
    var oDocument = FrameObject();

    oDocument.open("text/html");
    oDocument.write("<html>\n");
    oDocument.write("<head>\n");

    oDocument.write("</head>\n");
    oDocument.write('<body BGCOLOR="' + XDBBackgroundColor() + ' ' +
BACKGROUND="' + XDBBackgroundImage() + '">');
    oDocument.write(XD_sNewdoc);

    XDdisplayDiskInfo(oDocument);

    oDocument.write("</body>\n");
    oDocument.write("</html>\n");
    oDocument.close();
    XDrefreshAllButtons();
}

```

// parses the XML tree from the top frame and first calls show.

// This must be called on load of the main page.

function process(sExtra)

```

{
    if (XD_gsAction == '')
    {
        grove = Xparse(XD_gsXML);
    }
}

```

//this resets the variables that track how many files and folders
are selected

//don't reset if we are going into an action

XDresetSelected();

```

    }

    // If we have just performed an action that involved a folder then
    // we will open that folder so the user can see the results of the
    // action. To do so we update the old directory listing so that
    // the directory from which the action took place gets opened.
    if (XD_gnSelectedFolderID != '')
    {
        XD_gsPreviousGrove.index[XD_gnSelectedFolderID].attributes.show =
1;
    }

    // Now sync the view of the filesystem between the current and
    // previous views.
    synch(XD_gsPreviousGrove, grove);

    //reset attributes.selected for all items so that blue line does not
    get drawn
    XDresetAllSelected();
}

function BuildUpload()
{
    var oDocument = FrameObject();
    XD_gsActionUpload = true;
    HTMLGenericStart(oDocument);

    // var rand_num = parent.createRandomID();
    var rand_num = createRandomID();

    if (XD_gbExtraHelp)
    {
        oDocument.write(XDHelp(XD_gsHelpFileUpload));
    }

    oDocument.write("</TABLE>\n");

    oDocument.write('<p>\n');
    oDocument.write(XDHelp(XD_gsClientAd));
    oDocument.write('</p>\n');

    oDocument.write('<form name="form_upload" method="POST" action="/cgi-
bin/file_save.cgi" onSubmit="return
parent.parent.parent.openUpload(parent.parent.parent.XDCheckFormInput(),\'/cg
i-bin/file_upload_stat.cgi?id='+rand_num+\'\',\'window\',(this));"
TARGET="centerview"');

    // oDocument.write('<form name="form_upload" method="POST"
action="/cgi-bin/file_save.cgi" onSubmit="return
(parent.parent.parent.XDCheckFormInput());" TARGET="centerview"');

    oDocument.write(' enctype="multipart/form-data">'+ "\n");
    var results = '';
    results += '<input type="hidden" name="sFolderCurrent" value="'+
XDSelectedFolder() +'">\n';
    oDocument.write(results);

    oDocument.write('<input type=hidden name=id value='+rand_num+'>');

    oDocument.write('<TABLE cols=2>'+ "\n");

    for (var i=1; i<=5; i++)

```

```

    {
        oDocument.write('<tr><td valign="top" width="30"><FONT face="verdana,
arial, sans" size="-1"><b>' + XD_gsFile + i + ': </b></FONT></td><td><FONT
face="verdana, arial, sans" size="-1"><input type="file"
name="file_to_upload_0' + i + '" size="20"></FONT></td></tr>'+"\\n");
    }
    oDocument.write('</tr>'+"\\n");
    oDocument.write('<tr valign="top"> '+"\\n<td colspan=2>\\n");
    oDocument.write('<center>'+"\\n");
    oDocument.write(XDFormSubmitButtons());
    oDocument.write('</center>'+"\\n");
    oDocument.write('</td>'+"\\n");
    oDocument.write('</tr>'+"\\n");
    oDocument.write('</TABLE>');
    oDocument.write('</body>'+"\\n");
    oDocument.write('</html>'+"\\n");
    oDocument.close();
    XD_gnFrameHeight='85';
    return true;
}

```

```
function BuildCreate()
```

```

{
    var oDocument = FrameObject();
    HTMLGenericStart(oDocument);

    if (XD_gbExtraHelp)
    {
        oDocument.write(XDHelp(XD_gsHelpCreateFolder));
    }

    oDocument.write('<form name="form create" action="/cgi-
bin/folder_create.cgi" method="POST" onSubmit="return
parent.parent.parent.XDCheckFormInput();" target="centerview">');
    var results = '';
    results += '<input type="hidden" name="sFolderCurrent" value="' +
XDSelectedFolder() + '">\\n';
    oDocument.write(results);
    oDocument.write('<tr><td valign=center><B>' + XD_gsFolderName + ';</b>');
    oDocument.write('<input type="text" name="sFolderNew" value=""><br>');
    oDocument.write(XDFormSubmitButtons());
    oDocument.write('</td></tr>');

    oDocument.write('</TABLE>');
    oDocument.write('</body>'+"\\n");
    oDocument.write('</html>'+"\\n");
    oDocument.close();
    XD_gnFrameHeight='85';
    return true;
}

```

```
function BuildRename()
```

```

{
    var oDocument = FrameObject();
    HTMLGenericStart(oDocument);

    if (XD_gbExtraHelp)
    {
        oDocument.write(XDHelp(XD_gsHelpFolderRename));
    }
}

```

```

oDocument.write('<form method="POST" name="form_rename" action="/cgi-
bin/selected_rename.cgi" onSubmit="return
parent.parent.parent.XDCheckFormInput();"');
oDocument.write(' target="centerview" value="'+XDSelected()+'">\n');
var results = '';
results += '<input type="hidden" name="sFolderCurrent"
value="'+XDSelectedFolder()+'">\n';
oDocument.write(results);
oDocument.write('<tr><td valign=center><B>' + XD_gsNewName + ';</b>');
oDocument.write('<input type="hidden" name="sItemCurrent" value="'+
XDSelected()+'">\n');

if (XDProfileEditExtensions)
{
oDocument.write('<input type="text" name="sItemNew" value="" +
XDSelectedThingName()+'">\n');
oDocument.write('<input type="hidden" name="sItemExtension"
value="">\n');
}
else
{
oDocument.write('<input type="text" name="sItemNew"
value="'+XDSelectedThingNameMinusExtension()+'">'+XDSelectedThingNameExtensio
n()+'\n');
oDocument.write('<input type="hidden" name="sItemExtension"
value="'+XDSelectedThingNameExtension()+'">\n');
}
oDocument.write(XDFormSubmitButtons());
oDocument.write('</td></tr>');
oDocument.write('</TABLE>');
oDocument.write('</body>'+ "\n");
oDocument.write('</html>'+ "\n");
oDocument.close();
XD_gnFrameHeight='85';
return true;
}

function BuildDelete()
{
var oDocument = FrameObject();
HTMLGenericStart(oDocument);

var pathToFile = XDSelected();
var lastSlash = pathToFile.lastIndexOf('/');
var file = pathToFile.substring(lastSlash+1,pathToFile.length);

if (XD_gbExtraHelp)
{
oDocument.write(XDHelp(XD_gsHelpDelete));
}

oDocument.write('<form name="form_delete" action="/cgi-
bin/selected_delete.cgi" method="POST" onSubmit="return
parent.parent.parent.XDCheckFormInput();" target="centerview">');
var results = '';
results += '<input type="hidden" name="sFolderCurrent" value="'+
XDSelectedFolder()+'">\n';
oDocument.write(results);
oDocument.write('<tr><td valign=center><B>' + XD_gsSureDelete + ' ' +
file + ';</b>');
oDocument.write('<input type="hidden" name="sItemCurrent" value="" +
XDSelected()+ '"><br>');
oDocument.write('<input type="hidden" name="sFolderCurrent" value="" +
XDSelectedFolder()+ '"><br>');

```



```

oDocument.write(XDFormSubmitButtons());

oDocument.write('</td></tr>');
HTMLGenericEnd(oDocument);
XD_gnFrameHeight='85';
return true;
)

function BuildExplorer (grove,sStartDirectory)
{
    var returnValue = true;

    if (XD_gsAction == 'Upload')
    {
        returnValue = BuildUpload();
    }
    else if (XD_gsAction == 'Create')
    {
        returnValue = BuildCreate();
    }
    else if (XD_gsAction == 'Rename')
    {
        returnValue = BuildRename();
    }
    else if (XD_gsAction == 'Delete')
    {
        returnValue = BuildDelete();
    }
    else
    {
        var result = '';
        var nDepth = -2;

        result += '<TABLE compact border=0 cellpadding=4
width="'+ XD_gnExplorerTableWidth +' ">\n';
        result += XDFormSubmitButtons(1);

        result += "<tr><th align=\"left\">"+ XDEplorerFont() + '<font
size="2">' + XDPossessive(XD_gsFirstName + ' ' + XD_gsLastName) + " X:drive
<BR>"
        +XDDiskTotal()+" "+XD_gsCapacity+", "
        +XDDiskFree()+" "+XD_gsRemaining
        + "</th><th align=\"left\">"+

        XDEplorerFont()+'<font size="2">' + XD_gsSize + "</th><th
align=\"left\">"+
        XDEplorerFont()+'<font size="2">' + XD_gsLastModified +
"</th></tr>\n";
        result += dotag(grove, sStartDirectory, nDepth);
        result += "</TABLE>\n";
        XD_sNewdoc = result;

        show();

        //johngaa 11/22/99
        //Highlight bug fix
        if (XD_gsXOffset || XD_gsYOffset)
        {
            XD_goFrameFileExplorer.scrollTo(XD_gsXOffset,XD_gsYOffset);
        }
        //end of johngaa bug fix
    }
}

```

```

    }
    return returnValue;
}

function XDPossessive(name)
{
    var length = name.length;
    var lastChar = name.charAt(length-1);
    var possessive=name + "'s";
    if (lastChar == 's')
    {
        possessive = name + "'";
    }
    return possessive;
}

function XDExplorerFont()
{
    return '<font face="verdana, arial, sans">';
}

// constructs the HTML from the file explorer from the parsed XML
function dotag(tag, path, nDepth)
{
    path += '/' + tag.name;

    var result = '';
    var sCellColor = new String();
    var sIconImage = new String();
    var sFolderPointer = new String();
    var fileSize = new String();
    var fileString = new String();

    var sDate;    // The last modified date and time stamp

    //johngaa 11/23/99
    //highlight netscape bug fix
    // var sFlipFunction = new String('parent.parent.parent.flip(' +
tag.uid + ')');
    if (navigator.appName == "Netscape")
    {
        var sFlipFunction = new String('parent.parent.parent.flip(' +
tag.uid + ',window.pageXOffset,window.pageYOffset)');
    }
    else
    {
        var sFlipFunction = new String('parent.parent.parent.flip(' +
tag.uid + ',document.body.scrollLeft,document.body.scrollTop)');
    }

    //johngaa original 11/22/99
    //highlight netscape bug fix
    //var sSelectToggleFunction = new
String('parent.parent.parent.XDselectToggle(' + tag.uid + ')');
    if (navigator.appName == "Netscape")
    {
        var sSelectToggleFunction = new
String('parent.parent.parent.XDselectToggle(' + tag.uid +
',window.pageXOffset,window.pageYOffset)');
    }
}

```

```

        else
        {
            var sSelectToggleFunction = new
String('parent.parent.parent.XDselectToggle(' + tag.uid +
',document.body.scrollLeft,document.body.scrollTop)');
        }
        //end of johngaa bug fix

        // If the object is selected,
        // then add it to the selected arrays
        // and up the selected counts
        // and set the cell color to selected

        //set background color of the cells depending on status:  selected,
move or at rest
        if (tag.attributes.selected)
        {
            XD_gnSelectedCount=1;
            sCellColor =XD_gsSelectedColor;
            XD_gsSelectedList += PathRemovePrefix(path) + '+';

            if (tag.attributes.folder)
            {
                XD_gnSelectedFolderCount=1;
                XD_gnSelectedFolderID = tag.uid;
                XD_gsSelectedFolderList += PathRemovePrefix(path) + '+';
            }
            else
            {
                XD_gnSelectedFileCount=1;
            }
        }
        else if (tag.attributes.move)
        {
            // ELSE IF, it is set to move,
            // Then change the colors and
            sCellColor = XD_gsMoveSelectedColor;
        }
        else
        {
            // ELSE, set the cell color to not selected
            sCellColor = XD_gsNotSelectedColor;
        }

        if (tag.attributes.folder)
        {
            // SET special graphics and links for folder.
            nDepth++;

            if (tag.attributes.show)
            {
                if (tag.attributes.move)
                {
                    // The folder is open
                    sFolderPointer = '<IMG SRC="' +
XD_gimgOpenFolderPointer + '" BORDER="0">\n';
                    sIconImage = '<IMG SRC="' + XD_gimgOpenFolder + '"
BORDER="0" ALIGN="absmiddle" '+'\n\t'+ 'HSPACE="2" VSPACE="0" HEIGHT="16"
WIDTH="16">';
                }
                else
                {

```

```

        sFolderPointer = '<A HREF="javascript:' +
sFlipFunction + ';"><IMG SRC="' + XD_gimgOpenFolderPointer + '"
BORDER="0"></A>\n';
        sIconImage = '<IMG SRC="' + XD_gimgOpenFolder + '"
BORDER="0" ALIGN="absmiddle" '+'\n\t'+ 'HSPACE="2" VSPACE="0" HEIGHT="16"
WIDTH="16">';
    }
    }
    else
    {
        sFolderPointer = '<A HREF="javascript:' + sFlipFunction +
';"><IMG SRC="' + XD_gimgClosedFolderPointer + '" '+'\n\t'+ '
BORDER="0"></A>\n';
        sIconImage = '<IMG SRC="' + XD_gimgFolder + '" BORDER="0"
ALIGN="absmiddle" '+'\n\t'+ 'HSPACE="2" VSPACE="0" HEIGHT="16" WIDTH="16">';
    }
    }
    else
    {
        // This is a file and not a folder so show a FILE icon and do not
show any + or -
        sFolderPointer = ExplorerBlankFolderPointer();
        sIconImage = '<IMG SRC="' + XD_gimgFile + '" BORDER="0"
ALIGN="absmiddle" '+'\n\t'+ 'HSPACE="4" VSPACE="0">';
    }

    if (tag.attributes.size)
    {
        // SET file size indicator is attribute is present
        fileSize = XDExplorerFont()+tag.attributes.size+'k';
    }
    else
    {
        fileSize = '&nbsp;';
    }

    if (tag.attributes.lastModified)
    {
        sDate = tag.attributes.lastModified;
    }
    else
    {
        sDate = '&nbsp;';
    }

    if (tag.attributes.move)
    {
        fileString= sIconImage;
    }
    else
    {
        fileString = '<A HREF="javascript:' + sSelectToggleFunction +
';">'+ '\n' + sIconImage;
    }

    if ((tag.attributes.folder) || (!XDAction('Move')) ||
(tag.attributes.move))
    {
        // ONLY show IF it is (a folder or not in moving)
        // OR the object is question is being moved.
        result += '<A NAME="' + tag.name + '"></A><TR>';
    }

```

```

        result += '<TD BGCOLOR="' + sCellColor + '"
valign="absmiddle"><p>';
        result += "\n";
        result += "\n";
        result += _indent(nDepth);
        result += sFolderPointer;
        result += fileString;
        result += XDEplorerFont();
        result += '<FONT SIZE="2">';
        result += tag.name;
        result += '</A></TD>';
        result += "\n";
        result += "\n";
        result += '<TD BGCOLOR="' + sCellColor + '"
valign="absmiddle"><p><FONT SIZE="2">' + fileSize + '</FONT></TD>';
        result += '<TD BGCOLOR="' + sCellColor + '"
valign="absmiddle"><p><FONT SIZE="2">';
        result += XDEplorerFont();
        result += sDate;
        result += "</FONT></td>\n";

        result += '</TR>';
        result += "\n";
    }

    if (tag.attributes.show)
    {
        for (var i = 0; i < tag.contents.length; i++)
        {
            if (tag.contents[i].type == "element")
            {
                // To sort we simply recursively call ourselves with
the next element
                // in the sort order
                result += dotag(tag.contents[i], path, nDepth);
                result += "\n";
            }
        }
    }

    return result;
}

function ExplorerBlankFolderPointer ()
{
    return '<IMG SRC="/images/explorer/fnot.gif" WIDTH=15 HEIGHT=15
BORDER=0>\n';
}

// returns a true if the tag has any children that are selected
function XDopenChild(tag, children)
{
    var result = false;

    if (children)
    {
        if ((tag.attributes.selected) || (tag.attributes.move))
        {
            //added so user can close folder if items are selected
            //deselects item in folder if folder is closed
            tag.attributes.selected=false;

```



```

        }
        grove.index[id].attributes.show = 1;
    }

    BuildExplorer(grove, XD_gsRootDirectory);
}

// This is called when an item icon is clicked, causing
// it to toggle between selected and not selected

//original johngaa 11/22/99
//highlight netscape bug fix
//function XDselectToggle(id)
function XDselectToggle(id,xoffset,yoffset)
{

    //johngaa 11/22/99
    //highligt bug fix
    XD_gsYOffset = yoffset;
    XD_gsXOffset = xoffset;
    //end of johngaa bug fix

    // Martin to solve bug where we log in and we get the error grove.index
    // is not an object
    if (! grove.index)
    {
        return;
    }

    if (id>=0)
    {
        XDresetSelected();

        if (grove.index[id].attributes.selected)
        {
            grove.index[id].attributes.selected = false;
        }
        else
        {
            XDresetAllSelected();
            XD_gnSelectedCount++;
            grove.index[id].attributes.selected = true;
            if (grove.index[id].attributes.folder)
            {
                XD_gnSelectedFolderCount++;
                grove.index[id].attributes.show = 1;
            }
            else
            {
                XD_gnSelectedFileCount++;
            }
        }
    }
    else
    {
        XDresetAllSelected();
    }
    //if this is the page generated directly after a login
    //make XDrive the default and select it
    //then reset variable so we no longer select Xdrive as the default

```

```

        if (XD_gnLogin==1)
        {
            grove.index[0].attributes.selected=true;
            XD_gnLogin=0;
        }

        //this is called every time the file explorer changes
        //including creates, moves, deletes and renames
        //use a setTimeout for NS on NT because otherwise the
        //browser crashes if there is no wait period
        setTimeout("BuildExplorer(grove,XD_gsRootDirectory)",50);
        //BuildExplorer(grove,XD_gsRootDirectory);

    }

    // function to check to see if the root is selected
    function XDRootSelected()
    {
        if (grove.index[0].attributes.selected)
        {
            return true;
        }
        return false;
    }

    // This sets a selection to a value
    function XDselect (id,value)
    {
        // Martin to solve bug where we log in and we get the error grove.index
        // is not an object
        if (!grove.index)
        {
            return;
        }

        if (grove.index[id].attributes.folder)
        {
            grove.index[id].attributes.selected = true;
            grove.index[id].attributes.show = value;
        }
    }

    // DeSelects everything if so that only one thing can be selected
    // at a time, unless the the multipleSelect checkbox from myFrom
    // is selected.
    function XDresetAllSelected()
    {
        var length = grove.index.length;

        for (var i =0; i < length; i++)
        {
            grove.index[i].attributes.selected = 0;
        }
    }

    function XDresetAllMovedSelected()
    {
        // Martin bug fix -- after the first login could not show X:drive
        if (!grove.index)
    
```



```

    {
        return;
    }

    var length = grove.index.length;

    var oFrame = XD_goFrameUsageInfo;
    for (var i =0; i < length; i++)
    {
        grove.index[i].attributes.move = 0;
    }
}

// resets the number of selected, called by both flip and XDselectToggle
function XDresetSelected()
{
    XD_gsSelectedList = '';
    XD_gnSelectedCount = 0;
    XD_gnSelectedFolderCount =0;
    XD_gnSelectedFileCount =0;
    XD_gsSelectedFolderList = "";
}

function strip(str)
{
    var A = new Array();

    A = str.split("\n");
    str = A.join("");
    A = str.split(" ");
    str = A.join("");
    A = str.split("\t");
    str = A.join("");

    return str;
}

function entity(str)
{
    var A = new Array();

    A = str.split("&");
    str = A.join("&");
    A = str.split("<");
    str = A.join("<");
    A = str.split(">");
    str = A.join(">");

    return str;
}

function synch (prev_grove, new_grove)
{
    var prev_tag, new_tag, pi, ni;

    if (! prev_grove)
    {
        //set a flag so we know the first time a user logs in
        //there will be no prev_grove in this one case
        //flag is used to show blue bar on XDrive folder only right after
        logging in
        XD_gnLogin=1;
        return;
    }
}

```

```

    }
    //NS4.05 doesn't like this syntax
    //change to new syntax
    //if (! prev_grove.attributes)
    if (prev_grove.attributes!='')
    {
        return;
    }

    if (! new_grove.contents)
    {
        return;
    }

    if (prev_grove.attributes.show)
    {
        pi = 0;

        for (var ni = 0; ni < new_grove.contents.length; ni++)
        {
            if (new_grove.contents[ni].type == "element")
            {
                if (prev_grove.contents[pi])
                {
                    prev_tag = prev_grove.contents[pi];
                }

                if (new_grove.contents[ni])
                {
                    new_tag = new_grove.contents[ni];
                }

                if ((prev_tag) && (new_tag))
                {
                    if (prev_tag.name == new_tag.name)
                    {
                        // Make sure the contents for this object
                        // to avoid javascript "has no
                        // properties" errors.
                        if (prev_grove.contents[pi])
                        {
                            new_grove.contents[ni].attributes =
                                prev_grove.contents[pi].attributes;
                        }
                    }
                    else if (prev_tag.name > new_tag.name)
                    {
                        pi++;
                    }
                    else
                    {
                        ni++;
                    }
                }
            }
            synch(prev_grove.contents[pi],
                new_grove.contents[ni]);
        }

        pi++;
    }
}

```


//saveToXdrive.js

```

var win = external.menuArguments;

ExtMen = external.menuArguments;
ExtMenTag = ExtMen.event.srcElement;
ExtMenDoc = ExtMen.document;

var url;

function findAnchor(el) {

    while ((el!=null) && ((el.tagName!="A") && (el.href!="")))
        el = el.parentElement;
    return el;
}

function findUrl() {
    var re;
    var IMGinsideLink = false;

    //alert("Tag name is " + ExtMenTag.tagName);

    switch ( ExtMenTag.tagName ) {
        // if a "LINK", return the link's URL
        case "A" :
            url = ExtMenTag.href;
            break;

        case "TD":
            var el = win.document.selection.createRange();
            a = findAnchor(el.parentElement(0));
            if (a != null)
            {
                url = a.href;
            }
            break;

        // if it was an image, then this gets complicated:
        case "IMG" :

            // check all links to make sure we aren't in one:
            for ( count = 0; count < ExtMenDoc.links.length; count++ )
                if ( ExtMenDoc.links( count ).contains( ExtMenTag ) ) {
                    IMGinsideLink = true;
                    break;
                }

            // if none was found, return the image URL:
            if ( !IMGinsideLink )
                url = ExtMenTag.src;
            else {

                url = ExtMenDoc.links( count ).href;
            }
            break;
    }
}

```

```

default:
    url = ExtMenDoc.href;
    break;
}

// Replace "."
re = /%2e/g;
url = url.replace(re, ".");

// Replace ":"
re = /%3A/g;
url = url.replace(re, ".");

// See if from hotfiles ZD-Net
if (url.indexOf("hotfiles.zdnet") != -1)
{
    var startIndex;
    var endIndex;

    startIndex = url.indexOf("refresh_url=");
    if (startIndex != -1)
    {
        startIndex += 12;
        endIndex = url.indexOf("&", startIndex);

        if (endIndex != -1)
        {
            url = url.substring(startIndex, endIndex);
        }
    }
}

// see if from "download.com" C-Net
else if (url.indexOf("download.com") != -1)
{
    var indexHttp;
    var indexFtp;

    indexHttp = url.lastIndexOf("http://");
    indexFtp = url.lastIndexOf("ftp://");
    index = indexHttp;
    if (indexFtp > indexHttp)
        index = indexFtp;

    //alert( "index is " + index );

    if (index > 0)
    {
        var tempUrl;

        tempUrl = url.substr(index);
        url = tempUrl;
    }
}

}

findUrl();

//alert("begin");
//alert(url);

```

WO 01/33381

PCT/US00/30536

// Call X:Drive to perform actual copy

xd_skip(url);

//secure_login.js

```
//
// Written 12/1/99
// Description:
//   Allow users to login securely from the start
//
//
function getState()
{
    //
    //return the value of the checked item
    //called by checkSubmit
    //
    var state;
    if (document.Login.bSecurity[0].checked)
    {
        state = document.Login.bSecurity[0].value;
    }
    else
    {
        state = document.Login.bSecurity[1].value;
    }
    return state;
}

function checkSubmit()
{
    //
    // checks if secure toggle button is pressed or not
    // if it is don't allow the submission of the current
    // form but submit the secureLogin form
    //
    if (getState() == "on")
    {
        document.secureLogin.user.value = document.Login.user.value;
        document.secureLogin.pass.value = document.Login.pass.value;
        document.secureLogin.submit();
        return false;
    }
    else
    {
        return true;
    }
    return false;
}

function writeForm()
{
    //
    // creates a the secure form
    //
    var fullHostName = XDGetFullHostName();
    var cgiAction = "https://" + fullHostName + "/cgi-bin/login.cgi";
    var formStr;

    formStr = "<form name=\"secureLogin\" method=\"post\" action=\"";
    formStr += cgiAction;
    formStr += ">";
}
```

```
formStr += "<input type=\"hidden\" name=\"user\" value=\"\">";

formStr += "<input type=\"hidden\" name=\"pass\" value=\"\">";
formStr += "<input type=\"hidden\" name=\"bSecurity\"
value=\"on\">\n</form>";
document.writeln(formStr);

}

function clickSecureState()
{
    var tempL = new String(document.location);
    var start = -1;
    start = tempL.indexOf("https");
    if (start != -1)
    {
        if (document.Login.bSecurity[0].value == "on")
        {
            document.Login.bSecurity[0].click();
        }
        else
        {
            document.Login.bSecurity[1].click();
        }
    }
}
```


//skip.js

```

//*****
// xd_skip: Popup a skip the download window for the X:Drive skip
// the download service.
//
// Inputs:
//   file_url   : the absolute URL of the file to fetch
//   file_name  : the name to call the stored file
//   file_size  : the file size in KB
//
// Outputs:
//   none
//*****

var skipPartner;
var skipLanguage;
var height = 200;
var width = 575;

function xd_change_location (url)
{
    document.location=url;
}

function xd_skip(file_url,file_name,alt_url,catid,gid,sid,langauge,partner)
{
    (
        var base_url = "http://www.xdrive.com/cgi-bin/skip_the_download.cgi";

        if (! file_name || file_name.length == 0)
        {
            var ii;
            for (ii=0; ii<= file_url.length; ii++)
            {
                if (file_url.charAt(ii) == '/')
                {
                    file_name = '';
                }
                else
                {
                    file_name = file_name + file_url.charAt(ii);
                }
            }
        }

        var params = "FILEURL=" + escape(file_url) +
            "&FILENAME=" + escape(file_name) +
            "&ALTURL=" + escape(alt_url);

        if (langauge) {
            skipLanguage = langauge;
        }

        if (partner) {
            skipPartner = partner;
        }

        if (skipPartner)

```

```
{
    params = params + "&STDPARTNER=" + escape(skipPartner);
}

if (skipLanguage)
{
    params = params + "&LANG=" + escape(skipLanguage);
}

if (catid)
{
    params = params + "&CATID=" + escape(catid);
}

if (gid)
{
    params = params + "&GID=" + escape(gid);
}

if (sid)
{
    params = params + "&SID=" + escape(sid);
}

if(skipPartner == 'cnet')
{
    height = 235;
    width = 600;
}

url = base_url + "?" + params;
var d = new Date();
var name = d.getTime();

window.open
(
    url,
    name,

    'toolbar=no,menubar=no,scrollbars=no,fullscreen=no,resizable=no,width=' +
width + ',height=' + height
);

return;
}
```

//skipthedownload.js

```

<SCRIPT LANGUAGE="JavaScript"
SRC="http://www.xdrive.com/js/skip.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" defer>

var win = external.menuArguments;

ExtMen = external.menuArguments;
ExtMenTag = ExtMen.event.srcElement;
ExtMenDoc = ExtMen.document;

var url;

function findAnchor(el) {

    while ((el!=null) && ((el.tagName!="A") && (el.href!="")))
        el = el.parentElement;
    return el;
}

function findUrl() {
    var re;
    var IMGinsideLink = false;

    //alert("Tag name is " + ExtMenTag.tagName);

    switch ( ExtMenTag.tagName ) {
        // if a "LINK", return the link's URL
        case "A" :
            url = ExtMenTag.href;
            break;

        case "TD":
            var el = win.document.selection.createRange();
            a = findAnchor(el.parentElement(0));
            if (a != null)
            {
                url = a.href;
            }
            break;

        // if it was an image, then this gets complicated:
        case "IMG" :

            // check all links to make sure we aren't in one:
            for ( count = 0; count < ExtMenDoc.links.length; count++ )
                if ( ExtMenDoc.links( count ).contains( ExtMenTag ) ) {
                    IMGinsideLink = true;
                    break;
                }

            // if none was found, return the image URL:
            if ( !IMGinsideLink )
                url = ExtMenTag.src;
            else {

```

```

        url = ExtMenDoc.links( count ).href;
    }
    break;

default:
    url = ExtMenDoc.href;
    break;
}

// Replace "."
re = /%2e/g;
url = url.replace(re, ".");

// Replace ":"
re = /%3A/g;
url = url.replace(re, ".");

// See if from hotfiles ZD-Net
if (url.indexOf("hotfiles.zdnet") != -1)
{
    var startIndex;
    var endIndex;

    startIndex = url.indexOf("refresh_url=");
    if (startIndex != -1)
    {
        startIndex += 12;
        endIndex = url.indexOf("&", startIndex);

        if (endIndex != -1)
        {
            url = url.substring(startIndex, endIndex);
        }
    }
}

// see if from "download.com" C-Net
else if (url.indexOf("download.com") != -1)
{
    var indexHttp;
    var indexFtp;

    indexHttp = url.lastIndexOf("http://");
    indexFtp = url.lastIndexOf("ftp://");
    index = indexHttp;
    if (indexFtp > indexHttp)
        index = indexFtp;

    //alert( "index is " + index );

    if (index > 0)
    {
        var tempUrl;

        tempUrl = url.substr(index);
        url = tempUrl;
    }
}

}

findUrl();

```

```
//alert("begin");  
//alert(url);  
  
// Call X:Drive to perform actual copy  
xd_skip(url);  
  
</script>
```

//submit.js

```

/*****
*
* Submit.JS: This javascript class is for all the actions associated with
* buttons. This class may either open a new window or submit an existing
* form for server parsing.
*****/
/

/*****
* XDCheckFormInput() - check upload/rename/create input.
* if there are errors, give then alert. if not, submit
*****/

function XDCheckFormInput()
{
    //make sure user is not allowed to upload a blank file
    if (XD_gsAction == 'Upload')
    {
        sFormName = XD_goFrameFileExplorer.document.form_upload;
        if(sFormName.file_to_upload_01.value == '')
        {
            alert(XD_gsAlertUploadEmptyFile);
            return false;
        }
    }
    //make sure user cannot create a blank file
    else if (XD_gsAction == 'Create')
    {
        sFormName = XD_goFrameFileExplorer.document.form_create;
        if (sFormName.sFolderNew.value=='')
        {
            alert(XD_gsAlertCreateEmptyFile);
            return false;
        }
    }
    else if (XD_gsAction == 'Rename'),
    {
        sFormName = XD_goFrameFileExplorer.document.form_rename;
        //do not allow user to rename file the same name it already has
        //find just the file name to compare to what was input

        var lastSlash=sFormName.sItemCurrent.value.lastIndexOf('/');

        //if this is a folder of user may edit file extensions, use this
code
        if ((parent.parent.XDProfileEditExtensions) ||
(XD_gnSelectedFileCount==0))
        {
            //allow user to edit extensions so check everything after
the
            //last slash
            var
            fileName=sFormName.sItemCurrent.value.substring(lastSlash+1,sFormName.sItemCu
rrent.value.length);

            if (fileName == sFormName.sItemNew.value)

```

```

        {
            alert(XD_gsAlertRenameSameName);
            return false;
        }
    }
    else
    {
        //do not allow user to edit extensions so need to find last
        '..' as well
        var lastDot=sFormName.sItemCurrent.value.lastIndexOf('.');
        var
        fileName=sFormName.sItemCurrent.value.substring(lastSlash+1,lastDot);

        if (fileName == sFormName.sItemNew.value)
        {
            alert(XD_gsAlertRenameSameName);
            return false;
        }

        //check to see if user is trying to name the file/folder nothing
        //give em an error message if so
        if (sFormName.sItemNew.value == '')
        {
            alert(XD_gsAlertRenameNothing);
            return false;
        }
    }
    else { }

    XD_gsAction = '';
    //sFormName.submit();
    return true;
}

function XDSubmitView (sFormName) {
    // Always start by checking the status and if the status is not active
    then
    // return and do not perform any actions.
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_VIEW))

    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_VIEW)) {
        return false;
    }

    var sFixed = '';
    var sFileName = XDSelected();

    for (i = 0;i <= sFileName.length;i++) {
        if (sFileName.charAt(i) == ' ') {
            sFixed += '+';
        }
        else {
            sFixed += sFileName.charAt(i);
        }
    }

    // URL encode/escape string
    sFixed = escape(sFixed);

    var sURL = '/cgi-bin/file_load.cgi/'+sFixed+'?sFileCurrent=' + sFixed +
    "&source=www.fileExplorer.view";
}

```

```

        XDReaderShow(sURL, 400, 400);
        return true;
    }

    // Justin's upload status stuff.
    function openUpload(form_check, url, name, f) {

        if (! form_check) {
            return false;
        }

        var form_length = f.length;
        var cnt = 0;

        for(var i = 0; i < f.length; i++) {
            var e = f.elements[i];

            if ( (e.type == "file") && (e.value.length > 0) ) {
                cnt++;
            }
        }

        var amp_nof = "&nof=";
        url += amp_nof + cnt;

        msgWindow =
        window.open(url, name, 'width=350,height=190,toolbar=no,resize=no,scrollbars=no
        ');

        return true;
    }

    function createRandomID () {
        substr_rand_num = new String(Math.random());

        return substr_rand_num.substring(2,14);
    }

    function XDSubmitDownload ()
    {
        // Always start by checking the status and if the status is not
        // active then return and do not perform any actions.
        //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_DOWNLOAD))
        if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_DOWNLOAD))
        {
            return false;
        }

        var sFileName = XDSelected();
        var oDocument = XD_goFrameData.document;
        var sExtraPath;

        sExtraPath = '/' + sFileName;

        HTMLGenericStart(oDocument);
        oDocument.write('<form name="form_download" target="userData"
        method="POST" action="/cgi-bin/file_load.cgi' + sExtraPath + '"
        enctype="multipart/form-data">' + "\n");
        oDocument.write('<input type="hidden" name="sFileCurrent"
        value="' + sFileName + '">');
        oDocument.write('<input type="hidden" name="mime" value="download">');
        oDocument.write('<input type="hidden" name="source"
        value="www.fileExplorer.download">');
    }

```



```

    oDocument.write('<input type="hidden" name="sFolderCurrent" value="' +
XDSelectedFolder() + '">');
    //johngaa test add 12/2/99
    oDocument.write('</form>');
    //end of johngaa add
    HTMLGenericEnd(oDocument);
    oDocument.forms[0].submit();
    return true;
}

function XDSubmitNewFolder(sFormName)
{
    var sNewFolderName = prompt(XD_gsRenamePrompt);
    XDFormSetGeneric(sFormName);
    XDFormSetFolderNew(sFormName,sNewFolderName);
    sFormName.submit();
}

/*****
 * XDItemDelete: Delete an item (no prompting here)
*****/

function XDItemDelete()
{
    var sFileName = XDSelectedList();
    XDFormSetGeneric(sFormName);
    XDFormSetThingName(sFormName);
    sFormName.submit();
}

/*****
 * XDSubmitDelete: Verify they can delete the selected item and then
 * redirect to a web page that will prompt them to delete..
*****/

function XDSubmitDelete(sFormName)
{
    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertDeleteFolder);
        return false;
    }
    location = "delete_prompt.html";
    return true;
}

function XDBufferChange(sFormName,sType)
{
    // We popup a new window for them to select a folder from
    XDFormSetBufferAction(sFormName,sType);
    parent.parent.XDopenFolderSelectWindow();
    XDFormSetSelectedFiles(sFormName);
}

function XDSubmitBufferChange (sFolderTo)
{
    // This method is being access across frames so we cannot easily pass
the form name
    // so instead we set a variable equal to what the object would have
been.
    sFormName = window.frames[XD_gsControlFrame].document.form_buffer;

```

```

        XDFormSetGeneric(sFormName);
        XDFormSetFolderNew(sFormName, sFolderTo);
        sFormName.submit();
    }

function XDSubmitMove(sFormName)
{
    XDFrameMove();
    BuildExplorer(grove, XD_gsRootDirectory);
}

/*****
 * XDPopupShow: Show a popup browser
 *****/

function XDReaderShow(sURL, nHeight, nWidth) {

    nWidth = 500;
    nHeight = 600;

    var r = window.open(sURL, "reader", "location=no,toolbar=no,menubar=no,"+
        "status=no,resizable=yes,scrolling=yes,scrollbars=yes,"+
        "width="+nWidth+",height="+nHeight);

    /*** make sure the opener knows who the parent is
    if (r.opener == null) r.opener = self;

    /*** focus on the newly created window
    //r.focus();
    }

function FrameObject()
{
    return XD_goFrameFileExplorer.document;
}

function HTMLGenericStart (oDocument)
{
    oDocument.open("text/html");
    oDocument.write('<html>');
    oDocument.write("<head>\n");
    oDocument.write("<link rel=stylesheet href='/css/style_back.css'
type='text/css'>\n");
    oDocument.write("</head>\n");
    oDocument.write('<body background="' + XDBackgroundImage() + '"
bgcolor="' + XDBackgroundColor() + '">'+ "\n");
    oDocument.write('<table><tr>');
}

/*****
 * HTMLGenericEnd:
 *****/

function HTMLGenericEnd (oDocument)
{
    oDocument.write('</table>');
    oDocument.write('</body>'+ "\n");
    oDocument.write('</html>'+ "\n");
    oDocument.close();
}

```

```

function XDBuildForm()
{
    var form = '';
    var sSubmitButton = '/images/submit.gif';

    if (XDAction('Move'))
    {
        form += '<form name="form_buffer" action="/cgi-bin/buffer_paste.cgi" +
            ' method="POST" target="centerview"'+
            ' parent.parent.parent.XRReset();">'+"\\n";
        form += '<input type="hidden" name="sFile" value="' + XD_gsMoveSelectedList + '>';
        sSubmitButton = '/images/move.gif';
        XD_gnFrameHeight = '40';
    }

    form += '<input type="hidden" name="sFolderCurrent" value="' + XDSelectedFolder() + '>';
    form += '<input type="hidden" name="type" value="move">';
    form += '<input type="hidden" name="sItemCurrent" value="">';
    form += '<input type="hidden" name="sFolderNew" value="">';

    form += '<p><INPUT TYPE="button" VALUE="" + XD_gsButtonSubmit + '
onClick="parent.parent.parent.XRSetMoveForm(document.forms[0]);">' +
        '<INPUT TYPE="button" VALUE="" + XD_gsButtonCancel + '
onClick="parent.parent.parent.XRReset();
parent.parent.parent.XRRefreshExplorer();">' +
        '</td>';
    form += '</form>';

    return form;
}

function XDSetMoveForm (oForm)
{
    oForm.sItemCurrent.value = XDSelectedToMove();
    oForm.sFolderNew.value = XDSelectedFolder();

    // adding check for target folder
    if (XD_gsSelectedFolderList.length > 0)
    {
        //check to see if the user is attempting to move the file into
        //the folder it is already in - can't do that
        var slash=oForm.sItemCurrent.value.lastIndexOf("/");
        var fileDirectory=oForm.sItemCurrent.value.substring(0,slash);

        if (oForm.sFolderNew.value == fileDirectory)
        {
            alert(XD_gsAlertMoveSameFolder);
        }
        else
        {
            // makes sure that the target is not the same as
            // source
            if (oForm.sFolderNew.value == oForm.sItemCurrent.value)
            {
                alert(XD_gsAlertNoTargetFolder);
            }
            else
            {

```

```

the file      //call reset and submit form only if they can actually move
              //else they only get the dialog warning box
              XDReset();
              oForm.submit();
            }
          }
        else
        {
          alert(XD_gsAlertNoTargetFolder);
        }
      }
}

function XDFormSubmitButtons (generic)
{
  var HTMLString = '';
  var FormString = '';
  var TotalString = '';

  // Grab the appropriate HTML
  if (XDAction('Move'))
  {
    if (XD_gbExtraHelp)
    {
      HTMLString = XDHelp(XD_gsHelpMoveHTML);
    }

    FormString = XDBuildForm();
    return HTMLString + "</TD></TR><TR><TD>" + FormString;
  }
  else if (XDAction('Rename'))
  {
    if (XD_gbExtraHelp)
    {
      HTMLString = XDHelp(XD_gsHelpFolderRename);
    }
  }
  else if (XD_gbExtraHelp)
  {
    if (XD_gnSelectedCount > 0 && ! XDRootSelected())
    {
      if (XD_gnSelectedFileCount)
      {
        HTMLString = XD_gsHelpFileSelected;
      }
      else
      {
        HTMLString = XD_gsHelpFolderSelected;
      }
    }
    else
    {
      if (XD_gsFirstTime)
      {
        HTMLString = XD_gsHelpFirstTimeEnter;
      }
      else
      {
        HTMLString = XD_gsHelpEnter;
      }
    }
  }
}

```

```

        // Format the help box
        HTMLString = XDHelp(HTMLString);
    }

    if (! generic)
    {
        var sSubmitButton;
        sSubmitButton = '/images/submit.gif';
        if (XDAction('Rename'))
        {
            sSubmitButton = XD_gsButtonRename;
        }
        else if (XDAction('Upload'))
        {
            sSubmitButton = XD_gsButtonUpload;
        }
        else if (XDAction('Create'))
        {
            sSubmitButton = XD_gsButtonCreate;
        }
        else if (XDAction('Delete'))
        {
            sSubmitButton = XD_gsButtonDelete;
        }

        return '<p><input type="submit" value="' + sSubmitButton + '>\n' +
            '<input type=button value="' + XD_gsButtonCancel + '"
onclick="' +
            'parent.parent.parent.XRreset(); '+
            'parent.parent.parent.XDRefreshExplorer();">\n</FORM>';
    }

    TotalString = HTMLString + FormString;
    return TotalString;
}

function XDHelp (sHelp)
{
    return '<tr><td height=50 bgcolor="' + XD_gsExplorerHelpBackgroundColor
+ '"' colspan=3 valign=top><FONT FACE="arial, helvetica" size="-1"
color="#666666"><b>' + XD_gsInstructions + '</b>\n' + sHelp + '\n</td></tr>';
}

/*****
* XDFrameShare: Share a file with another user
*****/

function XDFrameShare()
{
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_SHARE))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_SHARE))
    {
        return false;
    }

    var sFile = XDEscapeCharacters(XDSelected());

```

```

frames['centerview'].document.location = '/cgi-
bin/share_a_file.cgi?help=' +
    XD_gbExtraHelp + '&sFileName=' + sFile;
return true;
}

function XDCheck (sName)
{
    return "if (! XDAllowChange("+sName+") {return false;}";
}

function XDSelectedThingName()
{
    var r = '';
    var s = XDSelected();
    for (var i=0; i<s.length;++i)
    {
        var ch=s.charAt(i);
        if (ch == '/')
        {
            r = '';
        }
        else
        {
            r += ch;
        }
    }
    return r;
}

function XDSelectedThingNameMinusExtension()
{
    var r = '';
    var b = false; // found first time
    var s = XDSelectedThingName();
    for (var i=s.length;i>=0;--i)
    {
        var ch=s.charAt(i);
        if (ch == '.' && ! b)
        {
            b = true;
            r = '';
        }
        else
        {
            r = ch + r;
        }
    }
    return r;
}

function XDSelectedThingNameExtension()
{
    var r = '';
    var s = XDSelectedThingName();
    var bFoundDot = false;
    for (var i=0;i<s.length;++i)
    {
        var ch = s.charAt(i);
        if (ch == '.')
        {
            r = '';
            bFoundDot = true;

```

```

        }
        else
        {
            r += ch;
        }
    }
    if (bFoundDot == true)
    {
        return '.'+r;
    }
    else
    {
        return '';
    }
}

/*****
* XDFrameUpload: Refresh the action frame with a form to perform the file
* upload and set the form values during the HTML creation itself.
*****/

function XDFrameUpload(sCurrentFolder)
{
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_UPLOAD))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_UPLOAD))
    {
        return false;
    }

    XDActionStart('Upload');
    XD_gnFrameHeight = '1';
    frames['centerview'].document.location = XDCenterView();
    return true;
}

function XDFrameFolderNew ()
{
    //if (!XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_NEWFOLDER))
    if (!XDfunctionStatus(XD_TOOLBAR_BUTTON_NEWFOLDER))
    {
        return false;
    }

    XDActionStart('Create');
    XD_gnFrameHeight = '1';
    frames['centerview'].document.location = XDCenterView();
    return true;
}

function XDFrameRename ()
{
    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertRenameFolder);
        return false;
    }
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_RENAME))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_RENAME))
    {
        return false;
    }
}

```

```

        XDActionStart('Rename');
        XD_gnFrameHeight = '1';
        frames['centerview'].document.location = XDCenterView();
        return true;
    }

function XDFrameDeletePrompt()
{
    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertDeleteFolder);
        return false;
    }

    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_DELETE))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_DELETE))
    {
        return false;
    }

    XDActionStart('Delete');
    XD_gnFrameHeight = '1';
    frames['centerview'].document.location = XDCenterView();
    return true;
}

/*****
 * XDsetSelectedToMove: takes all files that are currently selected and sets
 * their move attribute
*****/

function XDsetSelectedToMove(tag)
{
    if (tag.attributes.selected)
    {
        tag.attributes.selected = 0;
        tag.attributes.move = 1;
    }

    for (var i = 0; i < tag.contents.length; i++)
    {
        if (tag.contents[i].type == "element")
        {
            XDsetSelectedToMove(tag.contents[i]);
        }
    }
}

function XDFrameMove()
{
    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertMoveFolder);
        return false;
    }

    // XXX
    XD_gsMoveSelectedList = XD_gsSelectedList;
    XD_gsSelectedList = "";
    XDsetSelectedToMove(grove);

    XDActionStart('Move');
}

```



```

        XD_gnFrameHeight = '1';
        frames['centerview'].document.location = XDCenterView();

        return true;
    )

/*****
 * XDBrowserDownloadSupported: Returns true if the browser supports the
 *   download button. This includes all Netscape versions and IE 5 or later.
 *****/

function XDBrowserDownloadSupported()
{
    return !((navigator.appName == "Microsoft Internet Explorer") &&
        (parseInt(navigator.appVersion) <= 4 ));
}

function XDProfile(form)
{
    XDProfileEditExtensions = form.elements['bFileExtEdit'].checked;
    XD_gbExtraHelp = form.elements['bExtraHelp'].checked;
    XD_gbMarketing = form.elements['bMarketing'].checked;
    XD_gbNewsletter = form.elements['bNewsletter'].checked;
}

function XDLogout()
{
    var sUrl = '/cgi-bin/logout.cgi';
    parent.parent.location.href = sUrl;
}

/*****
 *
 * XDSelected: Return the currently selected file or folder and remove the
 *   plus that appears at then end -- used the separate elements in a multi
 *   file/folder list.
 *****/

function XDSelected()
{
    return XD_gsSelectedList.substring(0,XD_gsSelectedList.length-1);
}

function XDSelectedFolder()
{
    alert(XD_gsLengthofFolder + XD_gsSelectedFolderList.length);
    return
XD_gsSelectedFolderList.substring(0,XD_gsSelectedFolderList.length-1);
}

function XDSelectedToMove()
{
    return
XD_gsMoveSelectedList.substring(0,XD_gsMoveSelectedList.length-1);
}

/*****
 * XDCleanupPath: Cleanup the passed path by removing the "/X:drive/" prefix
 *   and the + postfix.
 *****/

```

```

function XDPathCleanup(sPath)
{
    var sCopy = sPath;
    sCopy = sCopy.substring(9,sCopy.length)
    //sCopy = sCopy.substring(0,sCopy.length-1);
    return sCopy;
}

function XDDomain ()
{
    baseAddress = java.net.InetAddress.getLocalHost();
    userDomain = baseAddress.getHostName();
    alert(userDomain.toString());
}

function XDXdrive ()
{
    XDDomain();
    return '/cgi-bin/explorer_user_data.cgi';
}

function XDCenterView ()
{
    return '/cgi-
bin/frame_generic.cgi?thtml=centerview.thtml&sFrameHeight=' +
XD_gnFrameHeight;
}

function XDReset ()
{
    XD_gnSelectedCount = 0;
    XD_gnSelectedFileCount = 0;
    XD_gnSelectedFolderCount = 0;
    XD_gsSelectedList = "";
    XD_gnSelectedFolderID = '';
    XD_gsMoveSelectedList = "";
    XD_gsSelectedFolderList = "";
    XD_gsTargetFolder = "";
    ControlsEnabled = true;
    XDresetAllMovedSelected();
    XDActionEnd();
}

function XDAllowChange (sFolder)
{
    if (sFolder == '' || sFolder == ' ' || sFolder == 'public' || sFolder
== 'private')
    {
        return false;
    }
    return true;
}

function XDAction (sAction)
{
    if (XD_gsAction == sAction)
    {
        return true;
    }
    return false;
}

```

```

// Register a new action
function XDActionStart (sAction)
{
    XD_gsAction = sAction;
}

// Clear the current action
function XDActionEnd ()
{
    XD_gsAction = '';
}

function XDRefreshExplorer()
{
    //reset the action before calling this function
    //or the action screen will be drawn
    XDActionEnd();
    XD_gnFrameHeight = '40';
    //also reset if a move has been started but never finished
    XDresetAllMovedSelected();
    frames['centerview'].document.location=XDCenterView();
}

function XDGetButtonFrameHeight(oDocument)
{
    oDocument.open("text/html");
    oDocument.write(XD_goButtonFrameHeight);
    oDocument.close;
}

function XDSetButtonFrameHeight(height)
{
    XD_gnButtonFrameHeight=height;
}

/*****
** XDRefreshBanner: Refresh the banner with a new advertisement.
*****/

function XDRefreshBanner()
{
    if (XDBannerOn())
    {
        frames['banner'].document.location = '/cgi-bin/ads.cgi';
        // WIP: parent 3 twice removed (from the above line)
    }
}

/*****
** XDBannerOn: Return true if we should display the banner.
*****/

function XDBannerOn()
{
    if (XD_gsPartner == 'xdrv')
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

```
function XDTellAFriend()
{
    var sUrl = '/cgi-
bin/tell_a_friend.cgi?numFriends='+XD_gnNumFriendsToTell;
    frames['centerview'].document.location=sUrl;
}

function XDAddSpace()
{
    var sUrl = '/cgi-bin/addspace.cgi?action=intro';
    frames['centerview'].document.location=sUrl;
}

function XDDownloadClient()
{
    var sUrl = '/cgi-bin/download_client.cgi';
    frames['centerview'].document.location=sUrl;
}
```

//uploadStatus.js

<!--

```
function openUpload(form_check, url, name, f) {
    if (! form_check) {
        return false;
    }

    var form_length = f.length;
    var cnt = 0;

    for(var i = 0; i < f.length; i++) {
        var e = f.elements[i];

        if ( (e.type == "file") && (e.value.length > 0) ) {
            cnt++;
        }
    }

    var amp_nof = "&nof=";
    url += amp_nof + cnt;

    msgWindow =
window.open(url,name,'width=350,height=190,toolbar=no,resize=no,scrollbars=no
');

    return true;
}

function createRandomID () {
    substr_rand_num = new String(Math.random());

    return substr_rand_num.substring(2,14);
}

. //-->
```

//utils.js

```

/*****
* XDFormSetThingName: Set the name for the thing in the passed form.
*****/

function XDFormSetThingName(sFormName)
{
    sFormName.sThingName.value = XDSelectedList();
}

function XDFormSetBufferAction(sFormName,sType)
{
    sFormName.type.value = sType;
}

function XDFormSetFolderCurrent(sFormName)
{
    sFormName.sFolderCurrent.value = XDSelectedFolder();
}

function XDFormSetSelectedFiles (sFormName)
{
    sFormName.sFile.value = XDSelectedList();
}

function XDFormSetFolderNew(sFormName,sFolderNameNew)
{
    sFormName.sFolderNew.value = sFolderNameNew;
}

/*****
* XDFormSetThingOld: Set the old name attribute for the rename CGI.
*****/

function XDFormSetThingOld(sFormName,sThingName)
{
    sFormName.sThingNameOld.value = sThingName;
}

/*****
* XDFormSetThingNew: Set the new name attribute for the rename CGI.
*****/

function XDFormSetThingNew(sFormName,sThingName)
{
    sFormName.sThingNameNew.value = sThingName;
}

function XDFormSetGeneric(sFormName)
{
    XDFormSetFolderCurrent(sFormName);
}

/*****
* XDPopupShow: Show a popup browser
*****/

function XDPopupShow(
    sURL, /*** (I) The URL to open in the popup window
    nHeight, /*** (I) The height of the popup

```

```

nWidth) //*** (I) The width of the popup
{
var w = window.open(sURL, "viewer", "location=no,toolbar=no,menubar=no,"+
    "status=no,resizable=yes,scrolling=yes,scrollbars=no,"+
    "width="+nWidth+",height="+nHeight);

//*** make sure the opener knows who the parent is
if (w.opener == null) w.opener = self;

//*** focus on the newly created window
w.focus();
}

function XDSelectedList()
{
return XD_gsSelectedList;
}

function XDBackgroundColor()
{
return XD_gsExplorerBackgroundColor;
}

function XDBackgroundImage()
{
return XD_gsBackgroundImage;
}

function XDSelectedFolder()
{
return
XD_gsSelectedFolderList.substring(0,XD_gsSelectedFolderList.length-1);
}

/*****
* XDCleanupPath: Cleanup the passed path by removing the "/X:drive/" prefix
* and the + postfix.
*****/
function XDPathCleanup(sPath)
{
var sCopy = sPath;
sCopy = sCopy.substring(9,sCopy.length)
    //sCopy = sCopy.substring(0,sCopy.length-1);
return sCopy;
}

function XDMultiSelect (sValue)
{
if (sValue != 'null' && sValue != "")
{
m_sMultiSelect = sValue;
}
else
{
return m_sMultiSelect;
}
}

function HTMLNavigation ()
{
var sHTML = HTMLStart()
    +'<table width="100%" border="0" cellpadding="0" cellspacing="0">';

```

```

        + '<tr align="left" valign="top" bgcolor="#5EB114">'
        + '<td></td>'
        + '</tr><tr>'
        + '<td></td>'
        + '</tr><tr>'
        + '<td></td>'
        + '</tr></table>'
        + '<a target="toolbar" href="http://www.mit.edu">MIT</a>'
        + '</BODY>\n</HTML>';
    return sHTML;
}

function HTMLStart ()
{
    return "<HTML>\n"
    + '<body bgcolor="#6961AB" topmargin="0" leftmargin="0" marginheight="0"
marginwidth="0" text="#FFFFFF" link="#FFFFFF" {onload}>'
    + "\n";
}

function HTMLEnd ()
{
    return "\n</BODY>\n</HTML>\n";
}

function RedrawToolBar()
{
    {
        var sWindow = 'window.toolbar';
        sWindow.document.write(HTMLStart()+'test'+HTMLEnd());
    }
}

function XDEscapeCharacters (str)
{
    {
        var A = new Array();

        A = str.split("+");
        str = A.join("%2B");

        A = str.split(" ");
        str = A.join("+");

        A = str.split("%");
        str = A.join("%25");

        A = str.split("&");
        str = A.join("%26");

        return str;
    }
}

```


//verify_lib.js

```

<!-- Begin Hiding from older browsers

/*****
***      Javascript library of functions commonly
***      used in HTML forms.
*****/

validateForm(form)
    attaches to the submit button and takes the
    form as an argument. Validates all the
    fields and will only let the form be submitted
    if all the fields validate.

checkForm()
    attaches to nothing. Is used by the script
    internally to allow compel() to function w/o
    calling alert(), which would cause an infinite
    loop.

requireElements(num)
    attaches to onLoad to initialize the array
    of required fields in the form.

addRequiredElements()
    attaches to nothing. Is used internally to
    construct a array of the names of all the
    required fields in a form. For this to work
    the form needs a "requiredElements" hidden
    input tag. It should be of this format:

    <INPUT TYPE="hidden" NAME="requiredElements" VALUE=" name:email:">

    List the required field names in order that
    they appear in the form. End each name with
    a ':' and lead the whole value with a blank
    space. If this tag is not used, then
    validateRequiredElements will identify a
    missing required field by its number in liue
    of the name.

compel(textfield)
    attaches to an onBlur event on a textfield.
    This causes focus to be kept on a textfield
    until checkForm() determines that they user
    has filled it out correctly.

required(textfield, num)
    attached to an onBlur of a field is required.
    The number is it's location on the
    required_elements array. i.e.

    <INPUT TYPE="text" NAME="name" onBlur="required(this, 0)">

    This tag declairs "name" as the first
    required field in the form.

validatePhone(textfield)
    attaches to an onChage of a textfield. This
    function validates to true only if the

```

textfield is blank or contains only
0-9, -, (, or)

validateEmail(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains an @

validateDate(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a date in
the format DD-MON-RRRR

validateDate_old(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a date in
the format DD/MM/YY

validateNum(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a number
between -1 and infinity

validateMoney(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a number
between with two decimal places (ie 2.56)

confirmDelete(textfield)
attaches to an onClick on a submit button
used as a delete button that you want the
user to confirm before engaging.

reset_b()
attaches to an onClick on a submit button.
If you have a reset button, used confirmDelete
or have a button that needs to override the
validity of the form, this function should
be attached to these buttons to allow them
to function.

*****/

```

emailOK = true;
phoneOK = true;
dateOK = true;
lengthOK = true;
all_numOK = true;
all_moneyOK = true;
deleteOK = true;
yearOK = true;
required_elements = new Array();
required_elements_names = new Array();
blurred = "";
in_required = false;
submitted = false;
var submitCount;

```

```

function validateForm(form) {
  addRequiredElements(form);
  if (!emailOK) {
    alert(XD_gsValidateEmail);
    return false;
  } else if (!phoneOK) {
    alert(XD_gsValidatePhone);
    return false;
  } else if (!dateOK) {
    alert(XD_gsValidateDate);
    return false;
  } else if (!lengthOK) {
    alert(XD_gsValidateLength);
    return false;
  } else if (!all_numOK) {
    alert(XD_gsValidateNumber);
    return false;
  } else if (!all_moneyOK) {
    alert(XD_gsValidateMoney);
    return false;
  } else if (!yearOK) {
    alert(XD_gsValidateYear);
    return false;
  } else if (!deleteOK) {
    return false;
  } else if (!validateRequiredElements()) {
    return false;
  }
}

```

```

function compel (textfield) {
  if (blurred == "") {
    blurred = textfield;
  }
  if (!checkForm()) {
    blurred.focus();
    blurred.select();
  }
  if (checkForm() && !in_required) {
    blurred = "";
  }
}

```

```

function checkForm() {
  if(!emailOK){
    return false;
  } else if(!phoneOK){
    return false;
  } else if(!dateOK){
    return false;
  } else if(!lengthOK){
    return false;
  } else if(!all_numOK){
    return false;
  } else if(!all_moneyOK){
    return false;
  } else if(!yearOK){
    return false;
  } else if(!deleteOK){
    return false;
  } else {
    return true;
  }
}

```

```

    }
}

function requireElements(num) {
    var i;
    for (i=0; i < num; i++) {
        required_elements[i] = false;
    }
}

function addRequiredElements(form) {
    var found = false;
    for (var n=0; n < form.length; n++) {
        if (form.elements[n].name == "requiredElements") {
            found = true;
        }
    }
    if (found) {
        var length = form.requiredElements.value.length;
        var start_index = 0;
        var end_index = 0;
        var num = 0;
        for (var i=0; i < length; i++) {
            var theChar = form.requiredElements.value.charAt(i);
            if (theChar == ":") {
                start_index = end_index + 1;
                end_index = i;
                var string =
form.requiredElements.value.substring(start_index, end_index);
                num = required_elements_names.length;
                required_elements_names[num] = string;

                } // end of if ":"
            } //end for loop
        } // end of found
    }

//check to see if the year is a 4-digit value greater than 1900
function validateYear(textfield)
{
    yearOK = true;

    //make sure the file contains only numbers
    for (var n=0; n < textfield.value.length; n++)
    {
        var theChar = textfield.value.charAt(n);
        if ((theChar >= "0") && (theChar <= "9"))
        {
            //do nothing, assume it's still true
        }
        else
        {
            //contains non numeric elements
            yearOK=false;
        }
    }

    if (!yearOK)
    {

```

```

        alert(XD_gsValidateContainNums);
    }

    if (textfield.value < 1900)
    {
        yearOK = false;
        alert(XD_gsValidateGreater1900);
    }

    if (textfield.value.length != 4)
    {
        yearOK = false;
        alert(XD_gsValidateFourDigits);
    }
}

// Checks for a properly formatted email
function validateEmail(textfield)
{
    emailOK = true;

    if ((textfield.value == "") || (textfield.value.indexOf("@") < 0))
    {
        emailOK = false;
        alert(XD_gsValidateEmailFormat);
        return false;
    }
    return true;
}

function required(textfield, num)
{
    var alert_show = false;
    in_required = true;
    if (blurred == "")
    {
        alert_show = true;
        blurred = textfield;
    }

    if(textfield.type == "select-one")
    {
        //if the first option is chosen, assume that is not a real
        //choice, simply a default
        if (textfield.selectedIndex == 0)
        {
            if (alert_show)
            {
                alert(XD_gsValidateField + textfield.name +
XD_gsValidateRequired);
            }
            blurred.focus();
            blurred.select();
            required_elements[num] = false;
        } //end if selectedIndex empty
        else if (textfield.selectedIndex > 0)
        {
            blurred = "";
            required_elements[num] = true;
            in_required = false;
        } //end else
    }
}

```

```

    } //end select-one

    if (textfield.type == "text" || textfield.type == "textarea" ||
    textfield.type == "password")
    {
        if(textfield.value.length==0)
        {
            if (alert_show)
            {
                alert(XD_gsValidateField + blurred.name +
XD_gsValidateRequired);
            } //end alert_show
            blurred.focus();
            blurred.select();
            required_elements[num] = false;
        } //end if length empty
        else if (textfield.value.length > 0)
        {
            blurred = "";
            required_elements[num] = true;
            in_required = false;
        } //end else
    } //end if text
}

function validateRequiredElements() {
    var length = required_elements.length;
    for (var i = 0; i < length; i++){
        if (!required_elements[i]){
            if (required_elements_names[i] == "") {
                alert(XD_gsValidateAllRequiredField + i +
XD_gsValidateNotFilled);
            }
            return false;
        } else {
            alert(required_elements_names[i] + XD_gsValidateNotFilled);
            return false;
        }
    } // end of false element
} // end of array
return true;
}

function validatePhone(textfield) {
    phoneOK=true;
    var digits = 0;

    //Number can only contains ten digits and proper characters
    for(var i = 0; i < textfield.value.length; i++) {
        var theChar = textfield.value.charAt(i);
        if ((theChar >= "0") && (theChar <= "9")) {
            digits++;
            continue;
        }

        if (theChar == " ") continue;
        if (theChar == "-") continue;
        if (theChar == "(") continue;
        if (theChar == ")") continue;

        //else

```

```

    phoneOK = false;
  } //end for

  phoneOK = phoneOK && (digits == 10);
  if (textfield.value == "") {
    phoneOK = true;
  }
  if (!phoneOK) {
    alert(XD_gsValidatePhoneFormat);
  }

  return phoneOK;
}

//Check that the date is in the form of DD-MON-YY
function validateDate(textfield) {
  dateOK=true;
  if ((textfield.value.charAt(0) > "3") || (textfield.value.charAt(0) <
"0")) {
    dateOK=false;
  }
  if ((textfield.value.charAt(1) > "9") || (textfield.value.charAt(0) <
"0")) {
    dateOK=false;
  }
  if ((textfield.value.charAt(7) > "9") || (textfield.value.charAt(7) <
"0")) {
    dateOK=false;
  }
  if ((textfield.value.charAt(8) > "9") || (textfield.value.charAt(8) <
"0")) {
    dateOK=false;
  }
  if ((textfield.value.charAt(9) > "9") || (textfield.value.charAt(9) <
"0")) {
    dateOK=false;
  }
  if ((textfield.value.charAt(10) > "9") || (textfield.value.charAt(10)
< "0")) {
    dateOK=false;
  }
  if (textfield.value.charAt(2) != "-") {
    dateOK=false;
  }
  if (textfield.value.charAt(6) != "-") {
    dateOK=false;
  }
  var month = textfield.value.substring(3, 6);
  month = month.toUpperCase();
  if (!(month == "JAN" || month == "FEB" ||
month == "MAR" || month == "APR" ||
month == "MAY" || month == "JUN" ||
month == "JUL" || month == "AUG" ||
month == "SEP" || month == "OCT" ||
month == "NOV" || month == "DEC")) {
    dateOK= false;
  }
  if (textfield.value == "") {
    dateOK = true;
  }
  if (!dateOK) {
    alert(XD_gsValidateDateFormat);
  }
}

```

```

    }
}

//Check that the date is in the form of DD/MM/YY
function validateDate_old(textfield) {
    dateOK=true;
    if ((textfield.value.charAt(0) > "9") || (textfield.value.charAt(0) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(1) > "9") || (textfield.value.charAt(1) <
"0")) {
        dateOK=false;
    }
    if (textfield.value.charAt(2) != "/"){
        dateOK=false;
    }
    if ((textfield.value.charAt(3) > "3") || (textfield.value.charAt(3) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(4) > "9") || (textfield.value.charAt(4) <
"0")) {
        dateOK=false;
    }
    if (textfield.value.charAt(5) != "/"){
        dateOK=false;
    }
    if ((textfield.value.charAt(6) > "9") || (textfield.value.charAt(6) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(7) > "9") || (textfield.value.charAt(7) <
"0")) {
        dateOK=false;
    }
    if (textfield.value == "") {
        dateOK = true;
    }
    if (!dateOK) {
        alert(XD_gsValidateDateFormat);
    }
}

```

// checks to see that the textfield contains only numbers

```

function validateNum(textfield)
{
    all_numOK = true;

    for (var i=0; i < textfield.value.length; i++)
    {
        var theChar = textfield.value.charAt(i);
        if ((theChar < "0") || (theChar > "9"))
        {
            if (textfield.value != "-1")
            {
                all_numOK = false;
                alert(XD_gsValidateContainNums);
                break;
            } // end of if not -
        }
    }
}

```



```

        } //end if not #
    } //end for

    return all_numOK;
}

// checks to see that the textfield contains two decimal places
function validateMoney(textfield) {

    all_moneyOK = true;

    for (var i=0; i < textfield.value.length; i++) {
        var theChar = textfield.value.charAt(i);
        if ((theChar < "0") || (theChar > "9")) {
            if (theChar != ".") {
                all_moneyOK = false;
                alert(XD_gsValidateMoneyFormat);
                break;
            }
        } //end if not #
    } //end for

    return all_moneyOK;
}

function validateLength(textfield, len)
{
    lengthOK = true;

    if (textfield.value.length < len)
    {
        lengthOK = false;
        alert(textfield.name + XD_gsValidateLengthFormat + len +
XD_gsValidateChars);
    }
}

// attache to delete buttons to confirm
function confirmDelete(textfield) {
    deleteOK = confirm(textfield.value + ": Are you sure?");
}

// attache to other buttons, such as add, to allow them to submit
// after a failed delete confirm
function reset_b() {
    deleteOK = true;
    emailOK = true;
    phoneOK = true;
    dateOK = true;
    lengthOK = true;
    yearOK = true;
    all_numOK = true;
    deleteOK = true;
    var length = required_elements.length;
    for (var i=0; i < length; i++) {
        required_elements[i] = true;
    }
}

// a function to error check with
function test() {
    alert("Testing!")
}

```

```

    }

    // checks to see that the textfield contains only numbers and is
    //between 13 and 16 characters in length
    function validateLengthandInput(textfield, minLength, maxLength, dateType)
    {
        all_numOK = true;

        if ((textfield.value.length<minLength) ||
            (textfield.value.length>maxLength))
        {
            all_numOK = false;

            if (minLength == maxLength)
            {
                if (dateType == "ExpDate") {
                    alert(XD_gsValidateExpDateFormat);
                }
                else {
                    alert(XD_gsValidateDateFormat + maxLength +
XD_gsValidateChars);
                }
            }
            else
            {
                alert(XD_gsValidateCard + minLength + XD_gsValidateAnd +
maxLength + XD_gsValidateChars);
            }

            return all_numOK;
        }

        for (var i=0; i < textfield.value.length; i++)
        {
            var theChar = textfield.value.charAt(i);
            if ((theChar < "0") || (theChar > "9"))
            {
                if (textfield.value != "-1")
                {
                    all_numOK = false;
                    alert(XD_gsValidateContainNums);
                    break;
                } // end of if not -
            } //end if not #
        } //end for

        return all_numOK;
    }

    function checkRequired(form)
    {
        var complete = true;
        var length = form.elements.length;

        addRequiredElements(form);
        for (var i=0; i<length; i++)
        {
            for (var j=0; j < required_elements_names.length; j++)
            {
                if (form.elements[i].name == required_elements_names[j])
                {

```

```

        if ((form.elements[i].type == "text") ||
            (form.elements[i].type == "password") ||
            (form.elements[i].type == "textarea"))
        {
            if (form.elements[i].value == '')
            {
                complete = false;
                break;
            }
        }
        else if (form.elements[i].type == "select-one")
        {
            if (form.elements[i].selectedIndex == 0)
            {
                complete = false;
                break;
            }
        }
        else
        {
            //don't worry about radio button
        }
    }

    if (!complete)
    {
        //Temp bug fix: could not read any variable from english_text.js
        (scope problem?)
        //was: alert(XD_gsValidateAllRequired)
        alert("A required field is not filled out. Please make sure
        all required fields are filled out before hitting submit.");
        return false;
    }
    else
    {
        /*
        Check if we've already submitted
        */

        if (!submitted)
        {
            submitted = true;
            submitCount = 0;
            //took this line out because it was breaking IE
            // and it's not used for submitting the form anyway
            //form.submit();
            return true;
        }
    }

    submitCount += 1;

    {
        var gender = "";
        var message = "";;
    }

```

```
        if (form.gender.value == "1" || form.gender.value == "2")
        {
            var gender;
            if (form.gender.value == "1")
            {
                gender = "Dude"
            }
            else
            {
                gender = "Lady"
            }
        }

        if (submitCount == 2)
        {
            message = " Hey " + gender + " give me a second while I send
info";
        }
        if (submitCount == 3)
        {
            message = "Okay... now your just pressing too much";
        }
        if (submitCount > 1  && submitCount < 4)
        {
            alert(message);
        }
    }

    return false;
}

function CheckPassword(form)
{
    var length = form.elements.length;
    var change=1;
    //Make sure passwords match
    if (form.elements[1].value !=
        form.elements[2].value)
    {
        alert(XD_gsValidatePasswords);
        change=0;
        return false;
    }

    if (change==1)
    {
        form.submit();
    }
    return true;
}
```

//xparse.js

```

function _element()
{
    this.type = "element";
    this.name = new String();
    this.attributes = new Array();
    this.contents = new Array();
    this.uid = _Xparse_count++;
    _Xparse_index[this.uid]=this;

    // Added by Martin Hald
    this.attributes.folder = 0;
}

function _chardata()
{
    this.type = "chardata";
    this.value = new String();
}

function _pi()
{
    this.type = "pi";
    this.value = new String();
}

function _comment()
{
    this.type = "comment";
    this.value = new String();
}

// an internal fragment that is passed between functions
function _frag()
{
    this.str = new String();
    this.ary = new Array();
    this.end = new String();
}

//////////
// global vars to track element UID's for the index
var _Xparse_count = 0;
var _Xparse_index = new Array();

//////////
//// Main public function that is called to
//// parse the XML string and return a root element object

function Xparse(src)
{
    // Hack added by Martin Hald to fix the grove[x] not an object error
    // where the grove object array indexes was shifted up by the previos
    // parsing
    _Xparse_count = 0;

    var frag = new _frag();

```

```

// remove bad \r characters and the prolog
frag.str = _prolog(src);

// create a root element to contain the document
var root = new _element();
root.name= XD_gsRootPath;
root.attributes.folder = 1;
root.attributes.show = 1;

// main recursive function to process the xml
frag = _compile(frag);

// all done, lets return the root element + index + document
root.contents = frag.ary;
root.index = _Xparse_index;
_Xparse_index = new Array();

return root;
}

//////////

//////////
//// transforms raw text input into a multilevel array
function _compile(frag)
{
    // keep circling and eating the str
    while (1)
    {
        // when the str is empty, return the fragment
        if (frag.str.length == 0)
        {
            return frag;
        }

        var TagStart = frag.str.indexOf("<");

        if (TagStart != 0)
        {
            // theres a chunk of characters here, store it and go on
            var thisary = frag.ary.length;
            frag.ary[thisary] = new _chardata();
            if (TagStart == -1)
            {
                frag.ary[thisary].value = _entity(frag.str);
                frag.str = "";
            }
            else
            {
                frag.ary[thisary].value =
                _entity(frag.str.substring(0, TagStart));
                frag.str = frag.str.substring(TagStart, frag.str.length);
            }
        }
        else
        {
            // determine what the next section is, and process it
            if (frag.str.substring(1,2) == "?")
            {
                frag = _tag_pi(frag);
            }
            else
        }
    }
}

```

```

    {
    if (frag.str.substring(1,4) == "!-")
    {
        frag = _tag_comment(frag);
    }
    else
    {
        if (frag.str.substring(1,9) == "![CDATA(")
        {
            frag = _tag_cdata(frag);
        }
        else
        {
            if (frag.str.substring(1,frag.end.length + 3) == "/"
+ frag.end + ">" || _remove_escapes(frag.str.substring(1,frag.end.length +
3)) == "/" + frag.end)
            {
                // found the end of the current tag, end the
recursive process and return
                frag.str = frag.str.substring(frag.end.length +
3,frag.str.length);
                frag.end = "";
                return frag;
            }
            else
            {
                frag = _tag_element(frag);
            }
        }
    }
    }
    }
    }
    return "";
}

```

```

//////////

```

```

//////////

```

```

///// functions to process different tags

```

```

function XDTrueSpaceIndex(frag)
{
    var length = frag.length;
    for (var i=0; i < length; i++)
    {
        if ( (frag.charAt(i) == " ")
            &&(frag.charAt(i-1) != "\\")
        )
        {
            break;
        }
    }

    return i;
}

```

```

function _tag_element(frag)
{

```

```

// initialize some temporary variables for manipulating the tag
var close = frag.str.indexOf(">");
var empty = (frag.str.substring(close - 1,close) == "/");
if (empty)
{
    close -= 1;
}

// split up the name and attributes
var starttag = _normalize(frag.str.substring(1,close));
//var nextspace = starttag.indexOf(" ");
var nextspace = XDTrueSpaceIndex(starttag);
var attribs = new String();
var name = new String();
if (nextspace != -1)
{
    name = starttag.substring(0,nextspace);
    attribs = starttag.substring(nextspace + 1,starttag.length);
}
else
{
    name = starttag;
}

var thisary = frag.ary.length;
frag.ary[thisary] = new _element();
frag.ary[thisary].name = _remove_escapes(name);

if (attribs.length > 0)
{
    frag.ary[thisary].attributes = _attribution(attribs);
}

if (!empty)
{
    // !!!! important,
    // take the contents of the tag and parse them
    var contents = new _frag();
    contents.str = frag.str.substring(close + 1,frag.str.length);
    contents.end = name;
    contents = _compile(contents);
    frag.ary[thisary].contents = contents.ary;
    frag.str = contents.str;
}
else
{
    frag.str = frag.str.substring(close + 2,frag.str.length);
}
return frag;
}

function _tag_pi(frag)
{
    var close = frag.str.indexOf("?>");
    var val = frag.str.substring(2,close);
    var thisary = frag.ary.length;
    frag.ary[thisary] = new _pi();
    frag.ary[thisary].value = val;
    frag.str = frag.str.substring(close + 2,frag.str.length);
    return frag;
}

function _tag_comment(frag)

```



```

    {
        var close = frag.str.indexOf("-->");
        var val = frag.str.substring(4,close);
        var thisary = frag.ary.length;
        frag.ary[thisary] = new _comment();
        frag.ary[thisary].value = val;
        frag.str = frag.str.substring(close + 3, frag.str.length);
        return frag;
    }

function _tag_cdata(frag)
    {
        var close = frag.str.indexOf("]]>");
        var val = frag.str.substring(9,close);
        var thisary = frag.ary.length;
        frag.ary[thisary] = new _chardata();
        frag.ary[thisary].value = val;
        frag.str = frag.str.substring(close + 3, frag.str.length);
        return frag;
    }

////////////////////////////////////

////////////////////////////////////
//// util for element attribute parsing
//// returns an array of all of the keys = values
function _attribution(str)
    {
        var all = new Array();
        while (1)
        {
            var eq = str.indexOf("=");
            if (str.length == 0 || eq == -1)
            {
                return all;
            }

            var id1 = str.indexOf("'");
            var id2 = str.indexOf("\"");
            var ids = new Number();
            var id = new String();
            if ((id1 < id2 && id1 != -1) || id2 == -1)
            {
                ids = id1;
                id = "'";
            }
            if ((id2 < id1 || id1 == -1) && id2 != -1)
            {
                ids = id2;
                id = "\"";
            }
            var nextid = str.indexOf(id, ids + 1);
            var val = str.substring(ids + 1, nextid);

            var name = xstrip(str.substring(0, eq));
            var entity = new String();
            entity = _entity(val);
            all[name] = entity;
            str = str.substring(nextid + 1, str.length);
        }
        return all;
    }

```

```
//////////
```

```
//////////
```

```
//// util to remove \r characters from input string
//// and return xml string without a prolog
```

```
function _prolog(str)
```

```
{
    var A = new Array();

    A = str.split("\r\n");
    str = A.join("\n");
    A = str.split("\r");
    str = A.join("\n");

    var start = str.indexOf("<");
    if (str.substring(start,start + 3) == "<?x" || str.substring(start,start
+ 3) == "<?X" )
    {
        var close = str.indexOf(">");
        str = str.substring(close + 2,str.length);
    }
    var start = str.indexOf("<!DOCTYPE");
    if (start != -1)
    {
        var close = str.indexOf(">",start) + 1;
        var dp = str.indexOf("[",start);
        if (dp < close && dp != -1)
        {
            close = str.indexOf("]>",start) + 2;
        }
        str = str.substring(close,str.length);
    }
    return str;
}
//////////
```

```
function _remove_escapes (str)
```

```
{
    var A = new Array();
    A = str.split("\\");
    str = A.join("");
    return str;
}
```

```
//////////
```

```
//// util to remove white characters from input string
```

```
function xstrip(str)
```

```
{
    A = str.split(" ");
    str = A.join("");
    A = str.split("\n");
    str = A.join("");
    A = str.split("\t");
    str = A.join("");

    //A = str.split(" ");
    //str = A.join("&nbsp;");
    //A = str.split("\n");
    //str = A.join("");
    //A = str.split(" ");
    //str = A.join("");
}
```

```

    //A = str.split("\t");
    //str = A.join("");

    return str;
}
//////////

//////////
//// util to replace white characters in input string
function _normalize(str)
{
    var A = new Array();

    A = str.split("\n");
    str = A.join(" ");
    A = str.split("\t");
    str = A.join(" ");

    return str;
}
//////////

//////////
//// util to replace internal entities in input string
function _entity(str)
{
    var A = new Array();

    //A = str.split("<");
    //str = A.join("<");
    //A = str.split(">");
    //str = A.join(">");
    //A = str.split(""");
    //str = A.join(""");
    //A = str.split("'");
    //str = A.join("'");

    //A = str.split("&");
    //str = A.join("&");

    //Get rid of any escapes
    A = str.split("\\");
    str = A.join("");

    return str;
}
//////////

```

CLAIMS

What is claimed is:

1. A shared computer network storage system, comprising:
 - a first database containing file data;
 - a second database containing information (metadata) about said file data of said first database;
 - a server, said server executing file commands on said first file database, said server contemporaneously updating said second metadatabase upon executing said file commands; and
 - a client application, said client application communicating with said server, said client application invoking file commands upon said server, said server executing said file commands and updating information regarding said first file and second metadata databases displayed by said client application; whereby

said client application controls files in said first file database and information regarding status of said first database files is more readily available by reference to said second metadatabase.
2. The shared computer network storage system of claim 1, wherein said first file database is distributed over at least two physical storage devices.
3. The shared computer network storage system of claim 1, wherein said second metadatabase is distributed over at least two physical storage devices.
4. The shared computer network storage system of claim 1, wherein said client application communicates with said server via a proxy.
5. The shared computer network storage system of claim 1, wherein said server comprises a non-routable network.
6. The shared computer network storage system of claim 1, wherein said server comprises a transaction processor.
7. The shared computer network storage system of claim 6, wherein said transaction processor guarantees access to and transactions on said first and second databases.
8. The shared computer network storage system of claim 1, wherein said server comprises an enterprise java bean cluster (EJBC).
9. The shared computer network storage system of claim 8, wherein said enterprise java bean cluster (EJBC) handles business logic and resource access methods as well as memory caching for common resources.
10. The shared computer network storage system of claim 1, wherein said server further comprises an application network.

11. The shared computer network storage system of claim 10, wherein said application network further
2 comprises a java application cluster.
12. The shared computer network storage system of claim 10, wherein said application network handles
2 display functions and resource requests.
13. The shared computer network storage system of claim 1, wherein said server further comprises a web
2 server.
14. The shared computer network storage system of claim 13, wherein said web server handles all requests
2 for static content and proxies requests for dynamic content.
15. The shared computer network storage system of claim 1, wherein said server further comprises a load
2 balancer, said load balancer proxying requests to a sub-server having the highest degree of availability or
functionality.
16. The shared computer network storage system of claim 1 wherein said server further comprises a DNS
2 redirector, said DNS redirector proxying requests to a resource having a highest degree of functionality.
17. The shared computer network storage system of claim 1 wherein said server further comprises:
2 a transaction processor, said transaction processor on a non-routable network, said transaction
processor guarantees access to and transactions on said first and second databases;
4 an enterprise java bean cluster (EJBC) on a non-routable network, said enterprise java bean cluster
(EJBC) coupled to said transaction processor and handling business logic and resource access methods a
6 well as memory caching for common resources;
8 an application network on a non-routable network, said application network coupled to said
enterprise java bean cluster, said application network including a java application cluster and handling
display functions and resource requests;
10 a web server, said web server coupled to said application network and handling all requests for
static content and proxies requests for dynamic content;
12 a load balancer, said load balancer coupled to said web server and proxying requests to a sub-
server having the highest degree of availability or functionality; and
14 a DNS redirector, said DNS redirector coupled to said load balancer and proxying requests to a
resource having a highest degree of functionality.
18. The shared computer network storage system of claim 1, wherein said client application is web-based.
19. The shared computer network storage system of claim 1, wherein said client application interacts with
2 an operating system running upon a computer upon which said client application is also running, said client
application adopting and implementing a visual display format similar to said operating system.

20. A shared computer network storage system, comprising:
- a first database containing file data, said first database distributed over at least two physical storage devices;
 - a second database containing information (metadata) about said file data of said first database, said second database distributed over at least two physical storage devices;
 - a server, said server executing file commands on said first file database, said server contemporaneously updating said second metadatabase upon executing said file commands, said server including:
 - a transaction processor, said transaction processor on a non-routable network, said transaction processor guarantees access to and transactions on said first and second databases;
 - an enterprise java bean cluster (EJBC) on a non-routable network, said enterprise java bean cluster (EJBC) coupled to said transaction processor and handling business logic and resource access methods as well as memory caching for common resources;
 - an application network on a non-routable network, said application network coupled to said enterprise java bean cluster, said application network including a java application cluster and handling display functions and resource requests;
 - a web server, said web server coupled to said application network and handling all requests for static content and proxies requests for dynamic content;
 - a load balancer, said load balancer coupled to said web server and proxying requests to a sub-server having the highest degree of availability or functionality; and
 - a DNS redirector, said DNS redirector coupled to said load balancer and proxying requests to a resource having a highest degree of functionality; and
 - a client application, said client application communicating with said server via a proxy, said client application invoking file commands upon said server, said server executing said file commands and updating information regarding said first file and second metadata databases displayed by said client application; whereby
- said client application controls files in said first file database and information regarding status of said first database files is more readily available by reference to said second metadatabase.

21. The shared computer network storage system of claim 20, wherein said client application is web-based.

22. The shared computer network storage system of claim 20, wherein said client application interacts with an operating system running upon a computer upon which said client application is also running, said client application adopting and implementing a visual display format similar to said operating system.

23. A method for providing private file space and information transfer over a public computer network, the steps comprising:
- providing a publicly-available private file space system coupled to the public computer network;
 - providing a client program in communication with the public computer network;
 - sending a request from said client program to said publicly-available private file space system

6 ("private system");
evaluating said request;
8 authenticating said request;
satisfying said request; and
10 returning a success indicator to said client program indicating the success or failure of said request;
whereby
12 said client program may create and control files held by said private system.

24. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 23, wherein the step of evaluating said request further comprises evaluating said request for
static content and returning an appropriate response if said request is for static content.

25. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 24, the steps further comprising:
providing an application network within said private system;
4 proxying said request to said application network; and
parsing a header of said request.

26. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 23, wherein said step of authenticating said request further comprises:
authenticating a user using said client program; and
4 authenticating said request made by said client program to ensure that it conforms with an account
associated with said user.

27. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 23, further comprising:
parsing multipart form data associated with said request;
4 determining said request's type; and
submitting said request.

28. A method for providing private file space and information transfer over a public computer network,
2 the steps comprising:
providing a publicly-available private file space system coupled to the public computer network;
4 providing a client program in communication with the public computer network;
sending a request from said client program to said publicly-available private file space system
6 ("private system");
evaluating said request for static content and returning an appropriate response if said request is for
8 static content;
providing an application network within said private system;
10 proxying said request to said application network; and
parsing a header of said request

12 authenticating said request by authenticating a user using said client program and authenticating
said request made by said client program to ensure that it conforms with an account associated with said
14 user;

parsing multipart form data associated with said request;

16 determining said request's type;

submitting said request;

18 satisfying said request; and

returning a success indicator to said client program indicating the success or failure of said request;

20 whereby

said client program may create and control files held by said private system.

29. A data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network, comprising:

a user data object;

4 a process request object; and

a recovery object;

6 said user information object, said process request object, and said recovery object associated within
a file action object.

30. The data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network as set forth in claim 29, wherein said user data object further comprises:

a user information object; and

4 a security object.

31. The data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network as set forth in claim 29, wherein said process request object further comprises:

4 a file operation object comprising said recovery object and a database IO object, a file IO object,
and an administration object.

32. The data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network as set forth in claim 29, wherein said recovery object further comprises:

a recovery IO object;

4 a mount status object;

a recovery administration object; and

6 a recovery process object.

33. A data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network, comprising:

a user data object, said user data object having a user information object; and a security object;

4 a process request object, said process request object including a file operation object, a database IO
object, a file IO object, and an administration object; and

6 a recovery object, said recovery object incorporated in said file operation object, said recovery
object including a recovery IO object, a mount status object, a recovery administration object, and a
8 recovery process object;

10 said user information object, said process request object, and said recovery object associated within
a file action object; whereby

file operations may be facilitated by the data structure including recovery from resource failure.

34. A shared file storage resource for a computer network, comprising:

2 an allocatable file storage resource;

a server, said server coupled to said storage resource, said server:

4 allocating individual user file space for a plurality of users on said storage resource;

receiving files for storage on said storage resource;

6 transmitting files stored on said storage resource;

generating control-protocol codes for transmitting said files;

8 receiving file commands for controlling files on said storage resource; and

10 transmitting display codes indicating file status on said storage resource, said display codes
representing said storage resource as a network drive;

12 a first network connection, said first network connection coupling said server to the computer
network;

a workstation, said workstation:

14 receiving files for storage on said storage resource;

transmitting files stored on said storage resource;

16 receiving file commands for controlling files on said storage resource; and

18 transmitting display codes indicating file status on said storage resource, said display codes
representing said storage resource as a network drive; whereby

20 a user may store, retrieve, and control files in a unique and secure file storage area on said
allocatable storage resource available throughout the computer network and detached from said
workstation.

35. The shared file storage resource for a computer network as set forth in claim 34, wherein said display
2 codes further comprise:

4 a browser-interpretable object, such as a JavaScript object, said object displaying file status on said
storage resource as a web page.

36. The shared file storage resource for a computer network as set forth in claim 34, further comprising:

2 a standalone program running on said workstation, said standalone program interpreting said
display codes and providing a seamless interface to said user, said seamless interface presenting said
4 storage resource as a local or network resource to said user and allowing said user to manipulate files on
said storage resource in the same manner as local storage resources such as a floppy disk drive or a local
6 hard drive.

37. The shared file storage resource for a computer network as set forth in claim 34, wherein said
2 computer network, further comprises:
the Internet.

38. A method for transferring data from a first network resource to a second network resource at the
2 direction of a user, the steps comprising:
submitting a first file location indicating data to be transferred to the second network resource;
4 the second network resource requesting said data at said first file location from the first network
resource;
6 the first network resource transmitting said data to the second network resource; and
the second network resource notifying the user of successful transfer upon successful reception of
8 said data; whereby
the user may use the first and second network resources to obtain and control said data.

39. The method for transferring data as set forth in claim 38, wherein the second network resource
2 comprises a subscriber-based system of network-available storage space.

40. The method for transferring data as set forth in claim 38, wherein the first and second network
2 resources are coupled to the Internet.

41. The method for transferring data as set forth in claim 38, the steps further comprising:
2 displaying to the user a status of transmission of said data from said first network resource to said
second network resource.

42. The method for transferring data as set forth in claim 38, the steps further comprising:
2 verifying the user as a subscriber to or member of the second network resource.

43. A method for transferring data from a first network resource to a second network resource at the
2 direction of a user, the steps comprising:
submitting a first file location indicating data to be transferred to the second network resource, the
4 second network resource being a subscriber-based system of network-available data storage space;
verifying the user as a subscriber to or member of the second network resource;
6 the second network resource requesting said data at said first file location from the first network
resource;
8 the first network resource transmitting said data to the second network resource via Internet;
displaying to the user a status of transmission of said data from said first network resource to said
10 second network resource; and
the second network resource notifying the user of successful transfer upon successful reception of
12 said data; whereby
the user may use the first and second network resources to obtain and control said data.

44. A client-server system for a network-based data storage and manipulation system, comprising:
a client system, said client system having a file access service and a file manipulation service;
a server, said server providing network-based data storage resources and responding to requests transmitted by said client system, said server effecting said requests;
said server determining if a client request is one for metadata regarding data stored upon said server;
said server providing said metadata if said client request is for metadata and transmitting said metadata to said file manipulation service; and
said server performing a file action if said client request is not for metadata, said server updating said metadata and transmitting said metadata to said file manipulation service; whereby
said server operates, and said client system presents, operations on said server in a manner similar to operations local to said client system.

45. The client-server system for a network-based data storage and manipulation system as set forth in claim 44, wherein said file access service further comprises:
a request processing layer for processing requests; and
a first network I/O layer for transmitting said requests to said server.

46. The client-server system for a network-based data storage and manipulation system as set forth in claim 44, wherein said file manipulation service further comprises:
a parser, said parser parsing said metadata from said server;
a data structure, said data structure receiving and preserving parsed data from said parser; and
a data display layer, said data display layer operating upon and displaying said parsed data;
whereby
metadata may be displayed to inform about data stored upon said server.

47. The client-server system for a network-based data storage and manipulation system as set forth in claim 46, wherein said parser is an XML parser.

48. The client-server system for a network-based data storage and manipulation system as set forth in claim 44, wherein said server further comprises:

a second network I/O layer, said second network I/O layer engaged when said requests are not for metadata, said second network I/O layer transmitting requests for file action; and
a resource access layer, said resource access layer receiving transmissions from said second network I/O layer and effecting said requests, said resource access layer engaged when said requests are for metadata, said resource access layer obtaining and transmitting said metadata; and
a metadata compiler, said metadata compiler receiving said metadata from said resource access layer, compiling said metadata, and transmitting said compiled metadata to said client system.

49. The client-server system for a network-based data storage and manipulation system as set forth in claim 48, wherein said metadata compiler is an XML generator.

50. A client-server system for a network-based data storage and manipulation system, comprising:

2 a client system, said client system having a file access service and a file manipulation service;

4 a server, said server providing network-based data storage resources, said server creating and maintaining metadata regarding stored data, said server responding to requests transmitted by said client system, said server effecting said requests;

6 said server determining if a client request is one for metadata;

8 said server providing said metadata if said client request is for metadata and transmitting said metadata to said file manipulation service;

10 said server performing a file action if said client request is not for metadata, said server updating said metadata and transmitting said metadata to said file manipulation service;

12 said file access service having a request processing layer for processing requests and a first network I/O layer for transmitting said requests to said server;

14 said file manipulation service having an XML parser, said XML parser parsing said metadata from said server, said file manipulation service having a data structure, said data structure receiving and preserving parsed data from said parser, and said file manipulation service having a data display layer, said data display layer operating upon and displaying said parsed data so that metadata may be displayed to inform about data stored upon said server; and

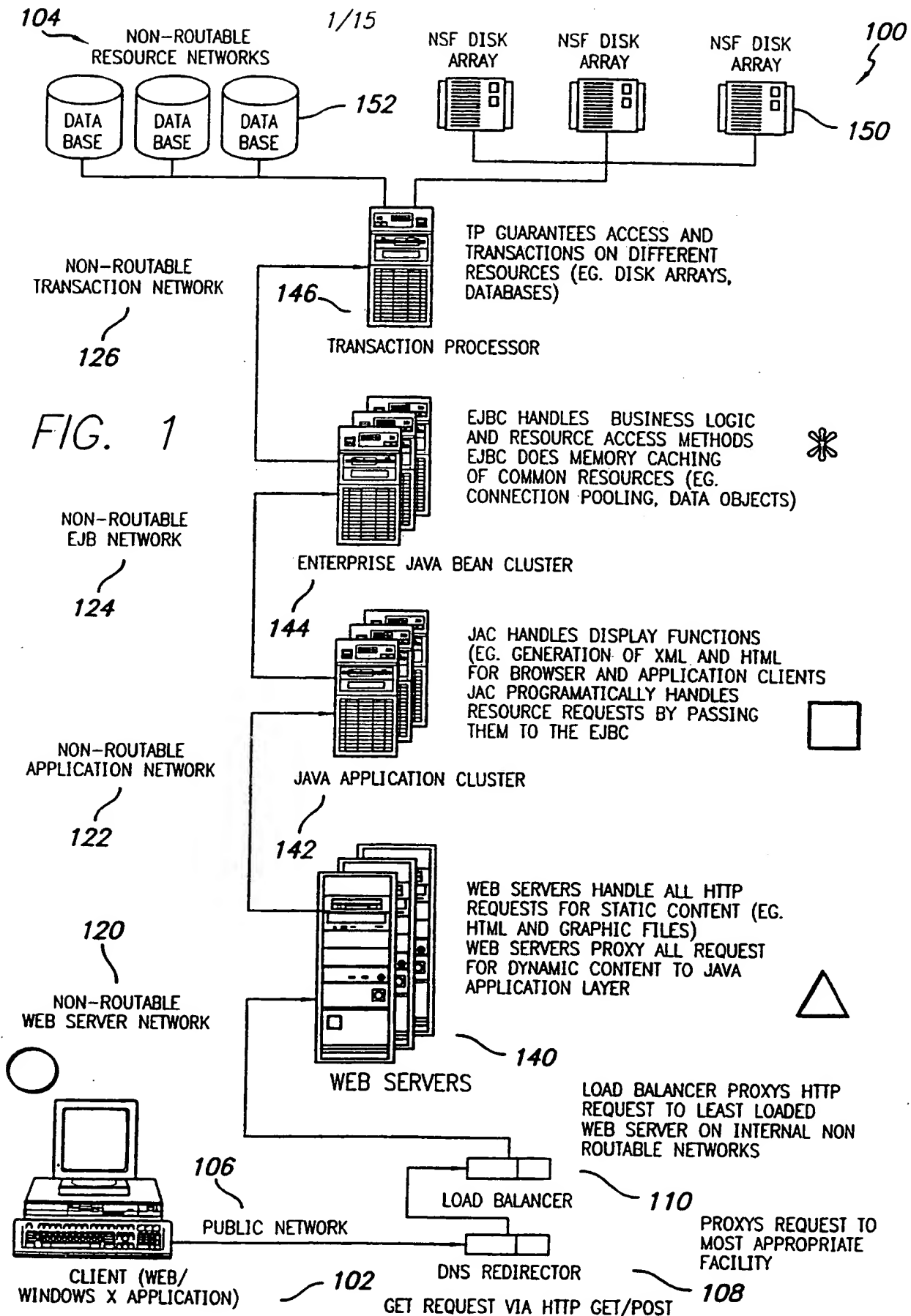
18 said server having a second network I/O layer, said second network I/O layer engaged when said requests are not for metadata, said second network I/O layer transmitting requests for file action, said server having a resource access layer, said resource access layer receiving transmissions from said second network I/O layer and effecting said requests, said resource access layer engaged when said requests are

20 for metadata, said resource access layer obtaining and transmitting said metadata, and said server having a metadata compiler in the form of an XML generator, said metadata compiler receiving said metadata from said resource access layer, compiling said metadata, and transmitting said compiled metadata to said client system; whereby

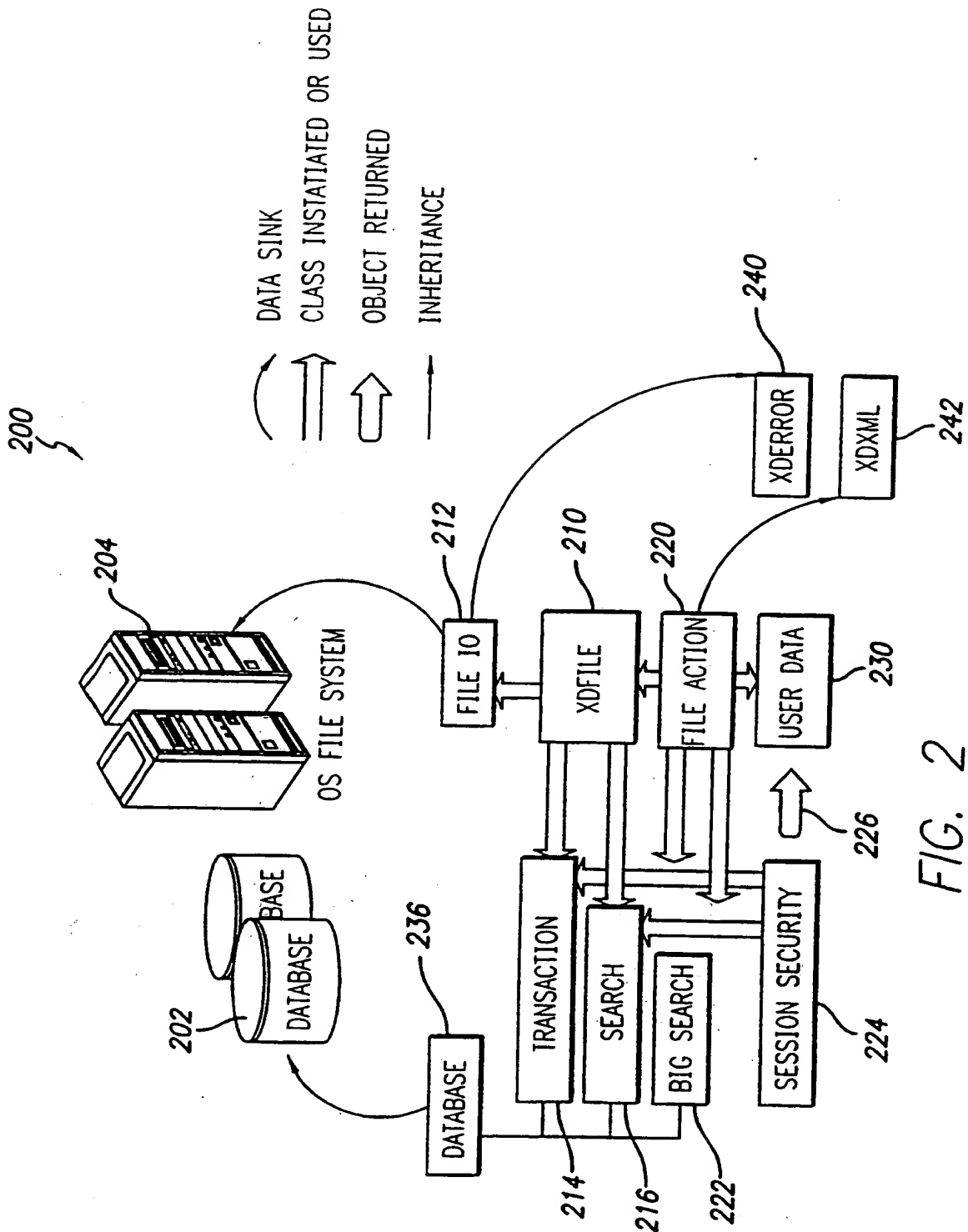
22

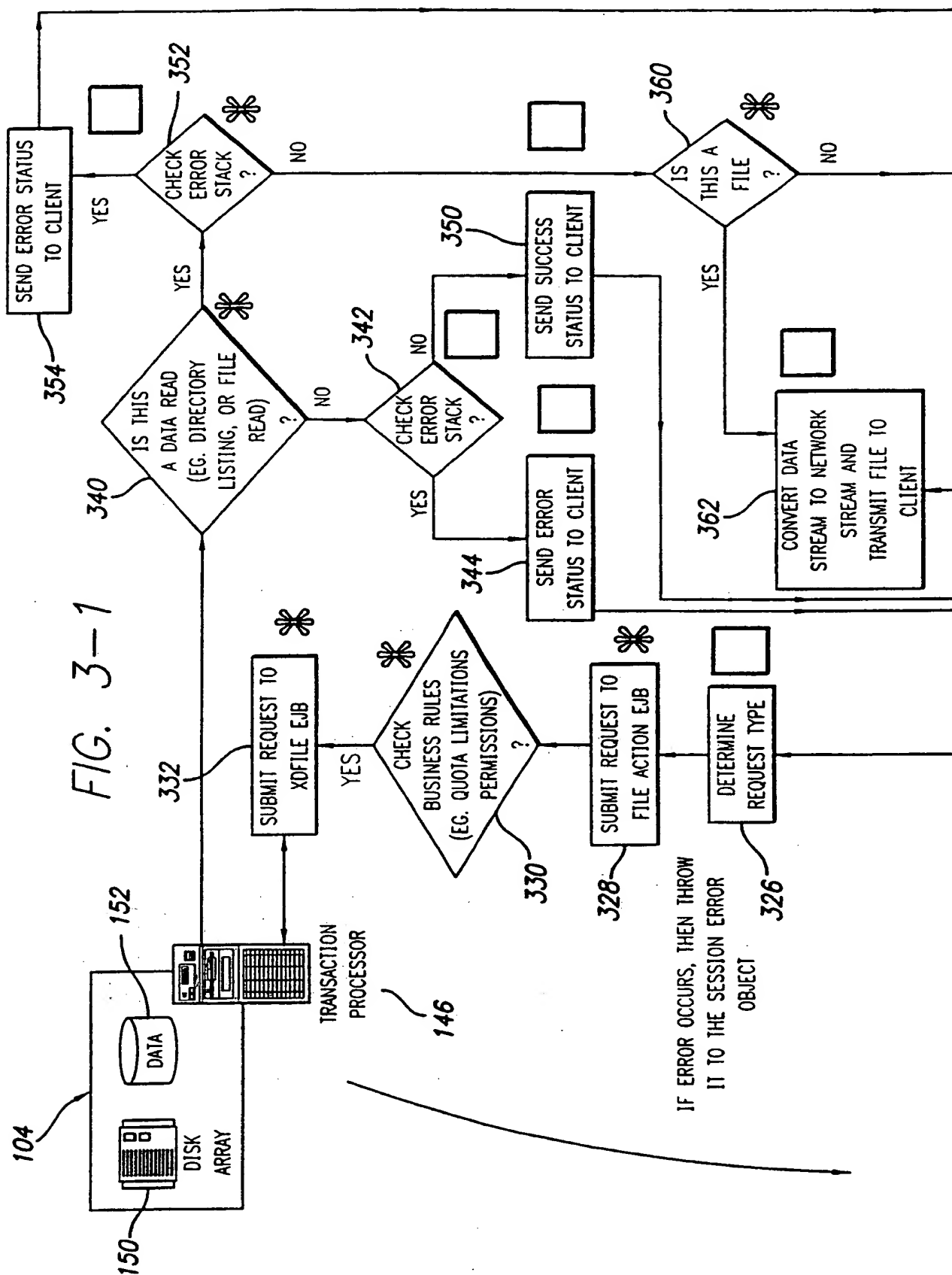
24

26 said server operates as and said client system presents operations on said server in a manner similar to operations local to said client system.



SUBSTITUTE SHEET (RULE 26)





4/15

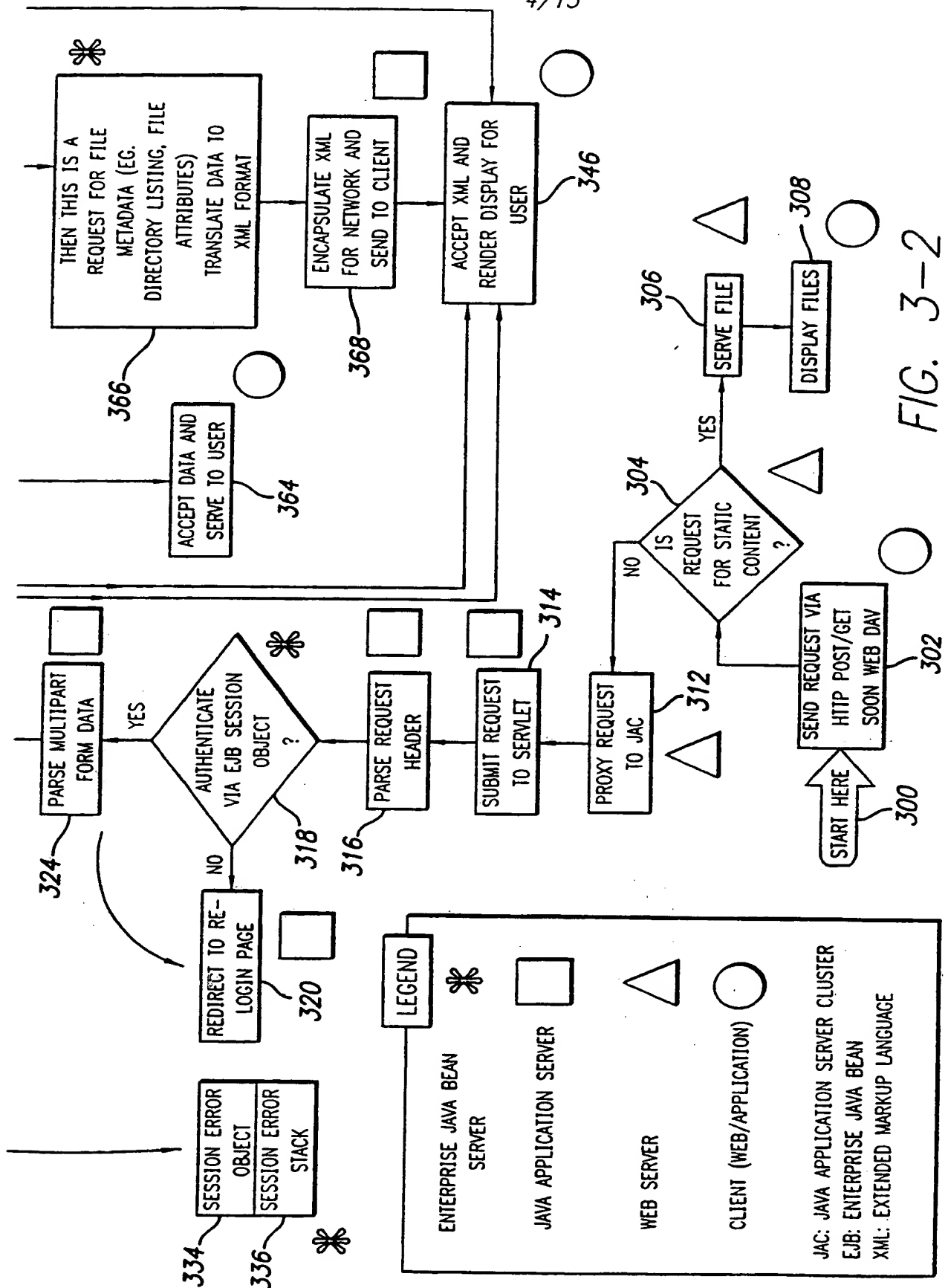
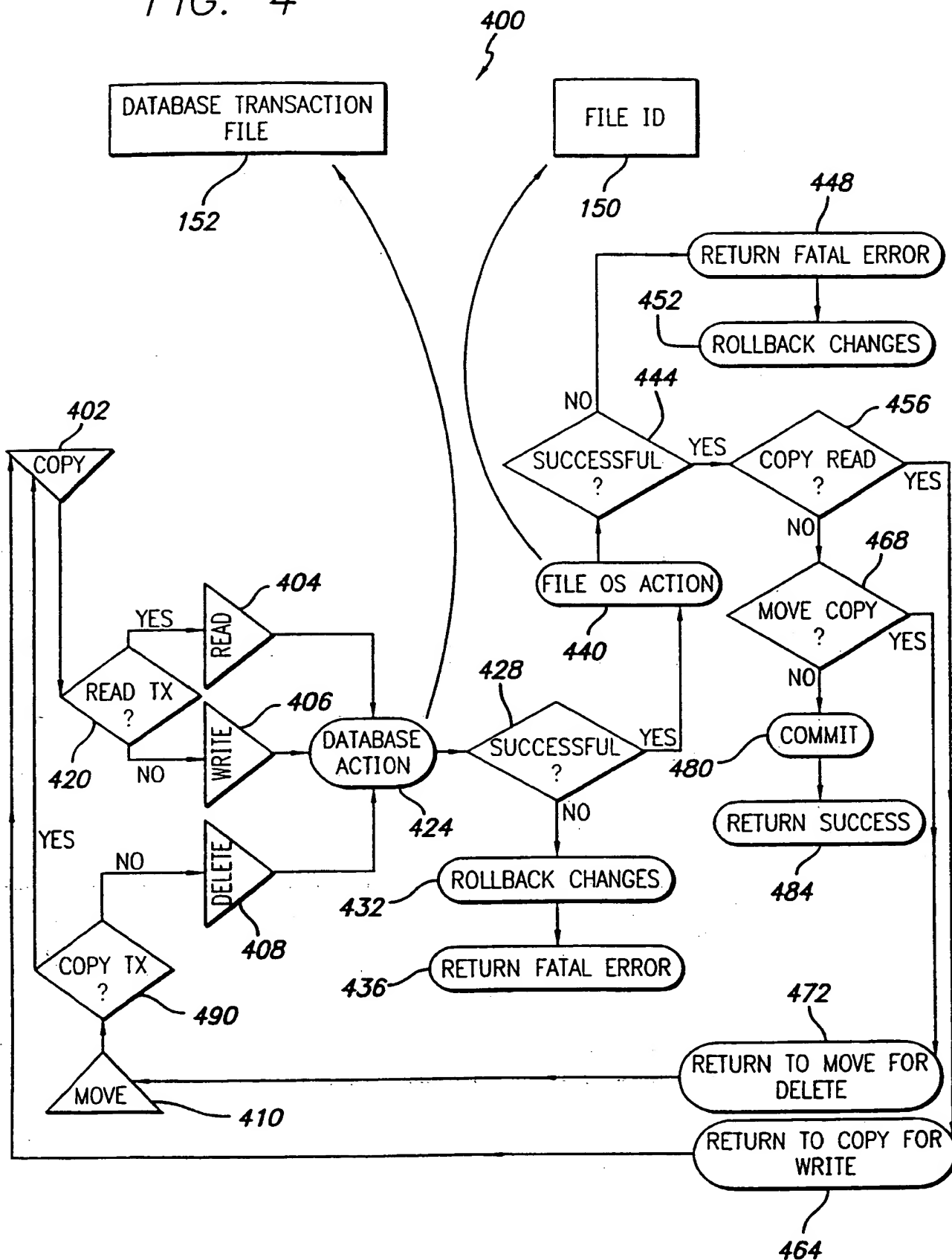


FIG. 3-2

FIG. 4

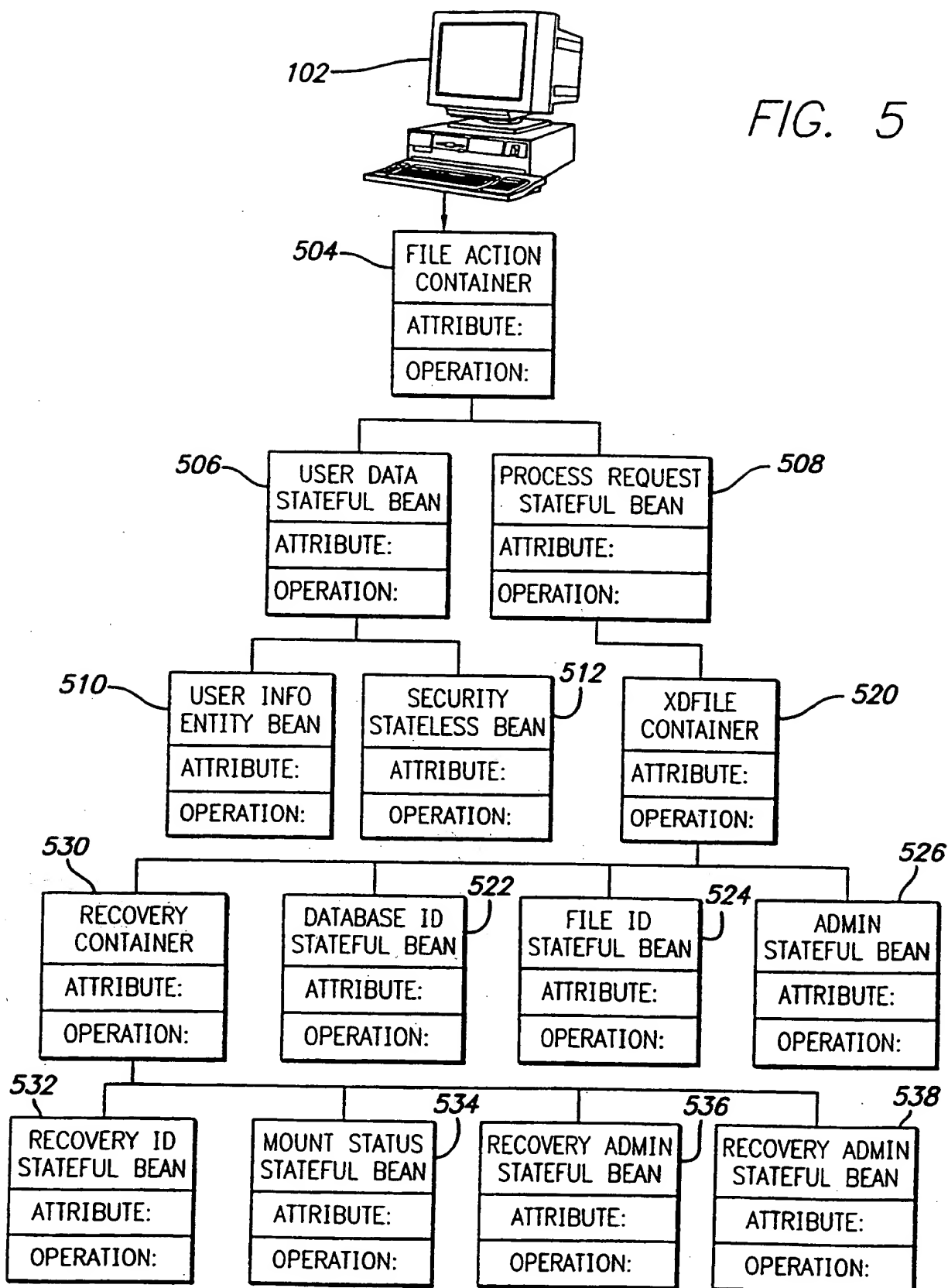
5/15



6/15

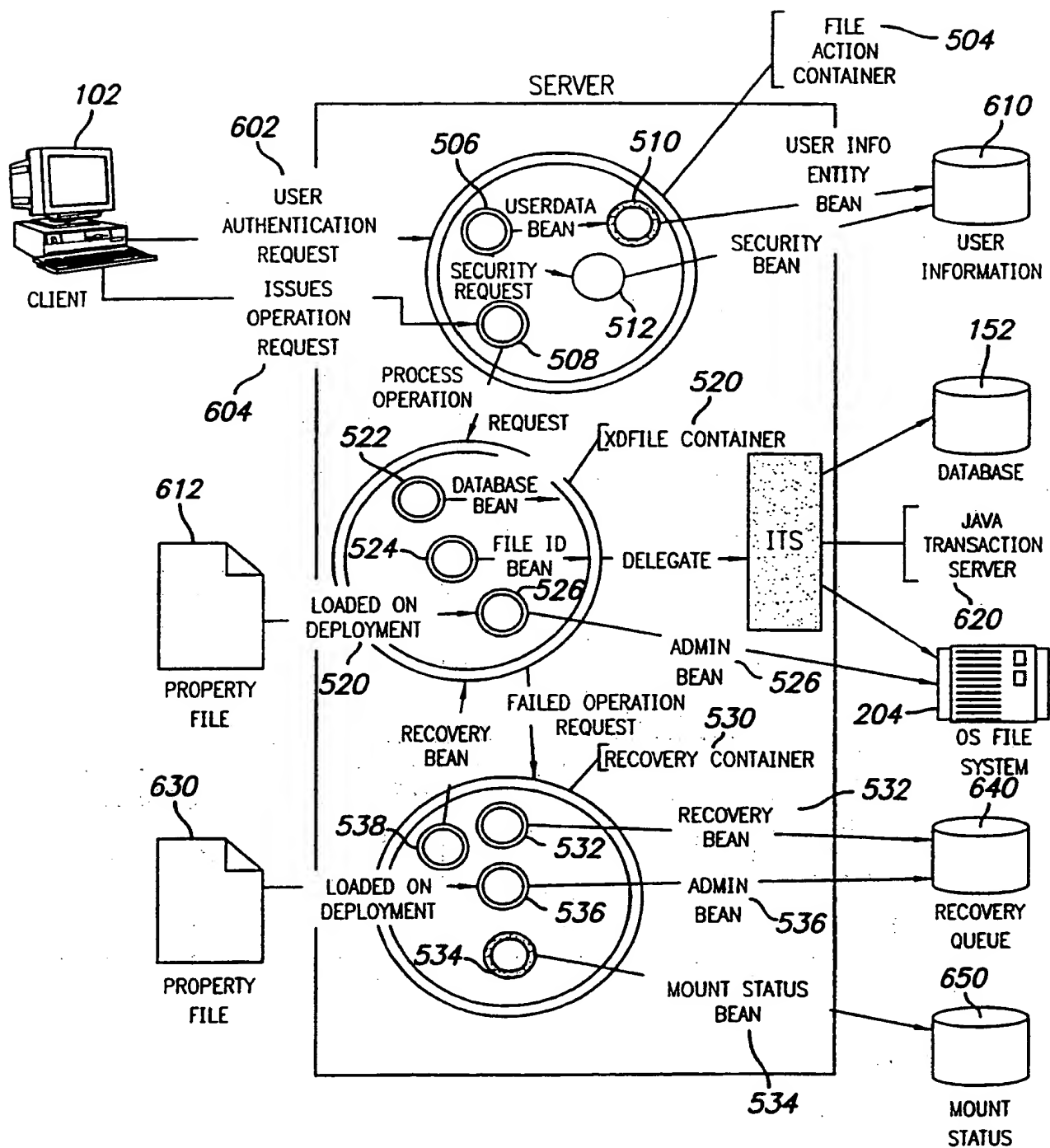
500

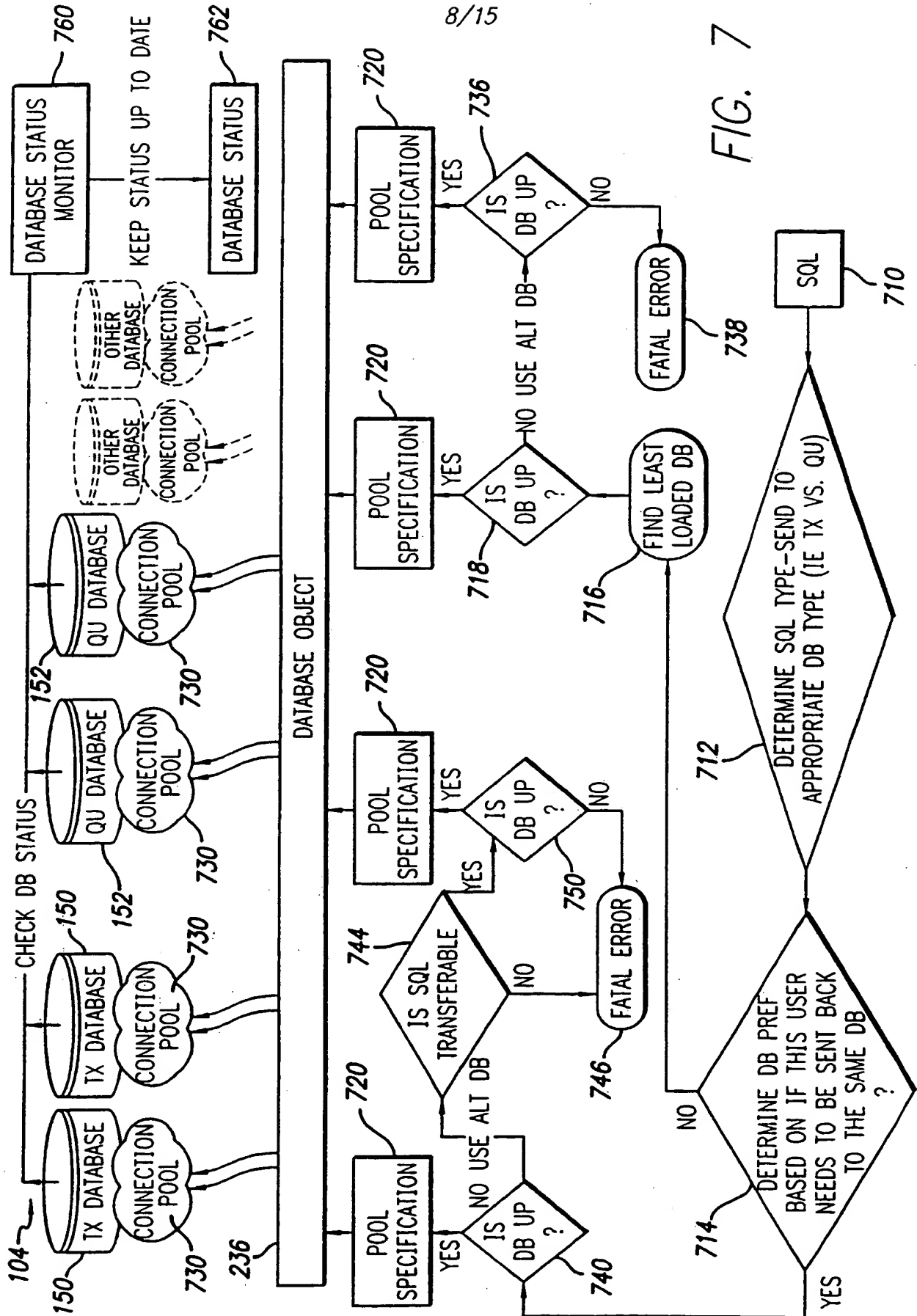
FIG. 5

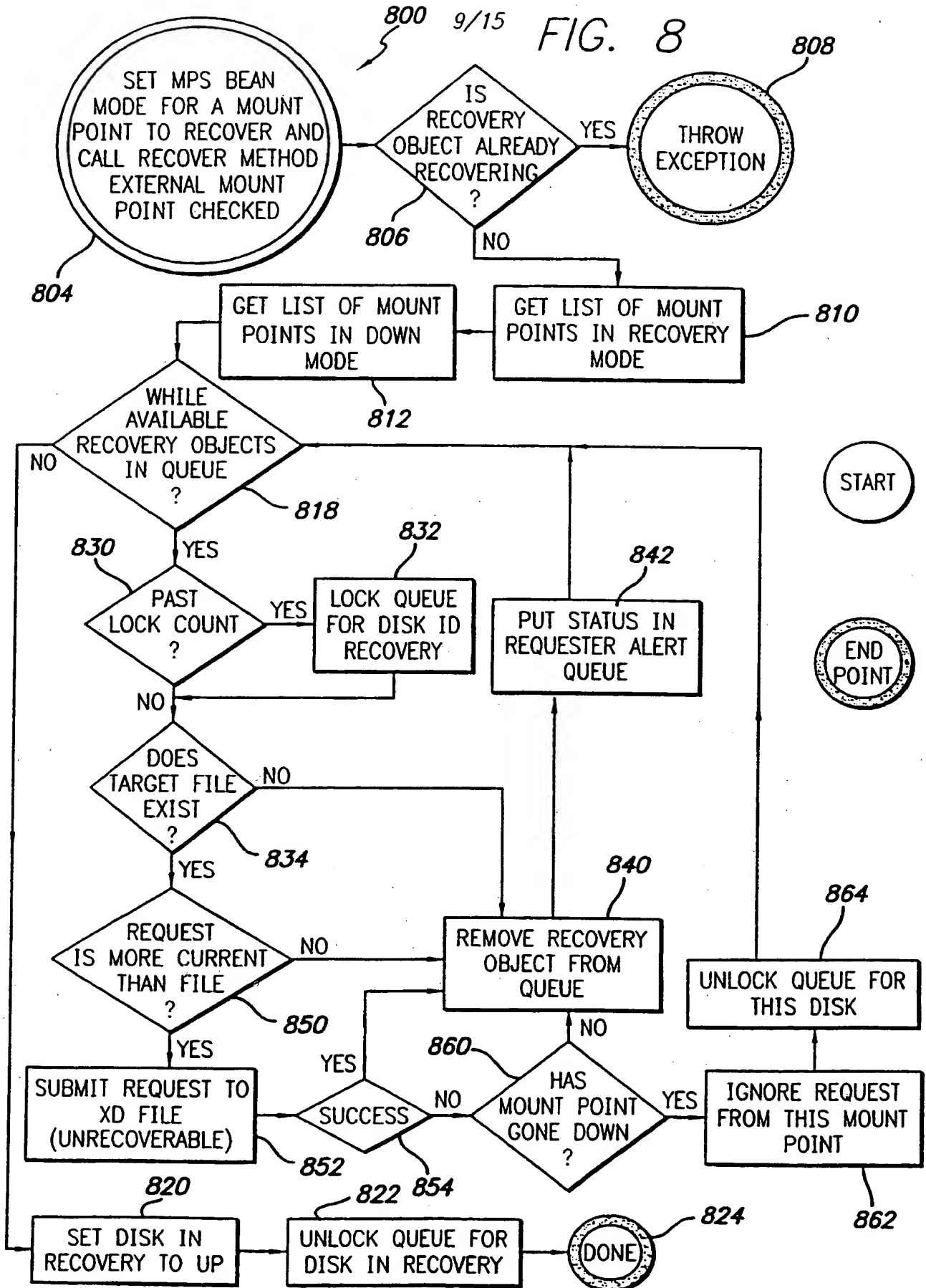


7/15

FIG. 6







SUBSTITUTE SHEET (RULE 26)

10/15

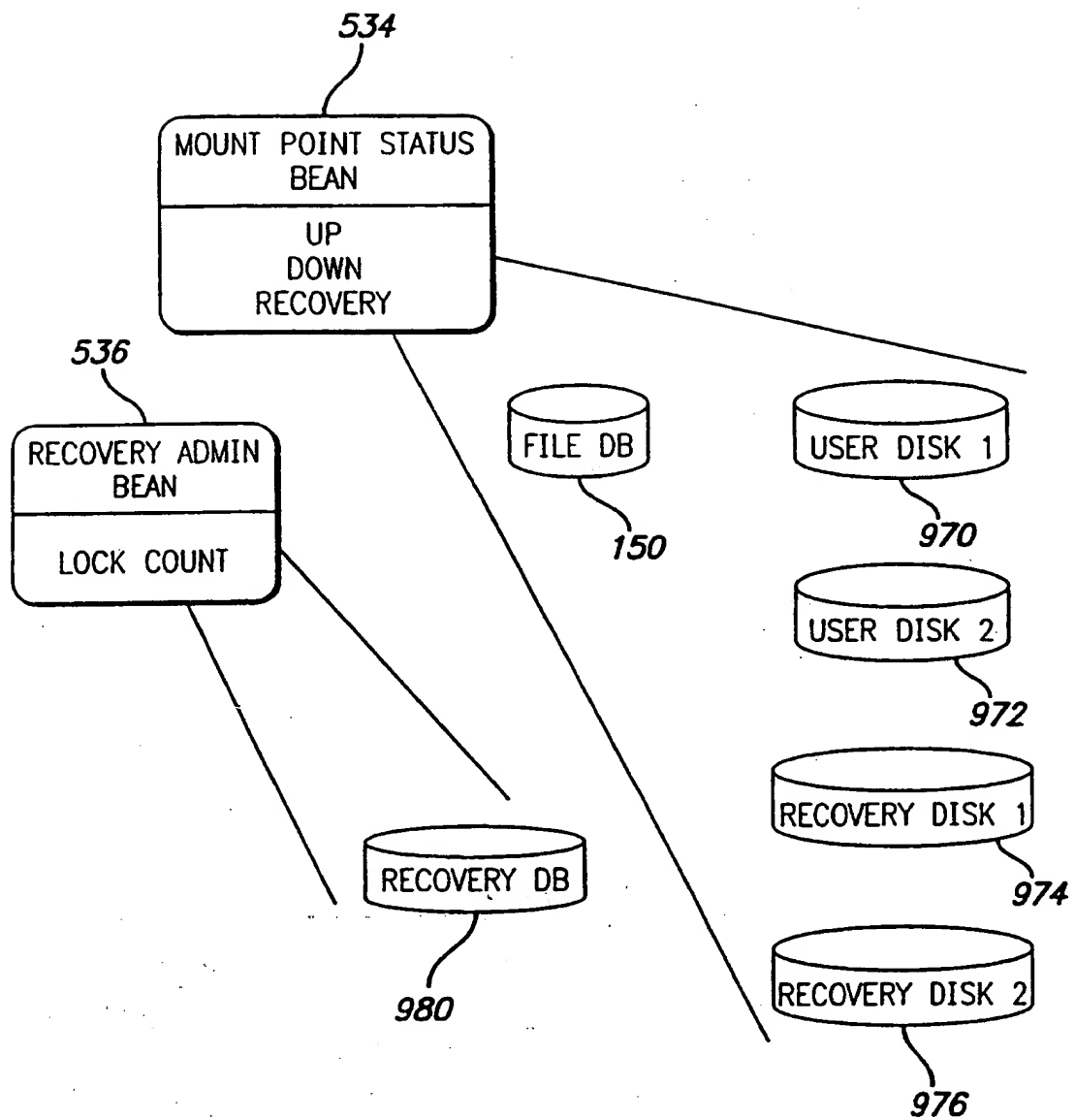
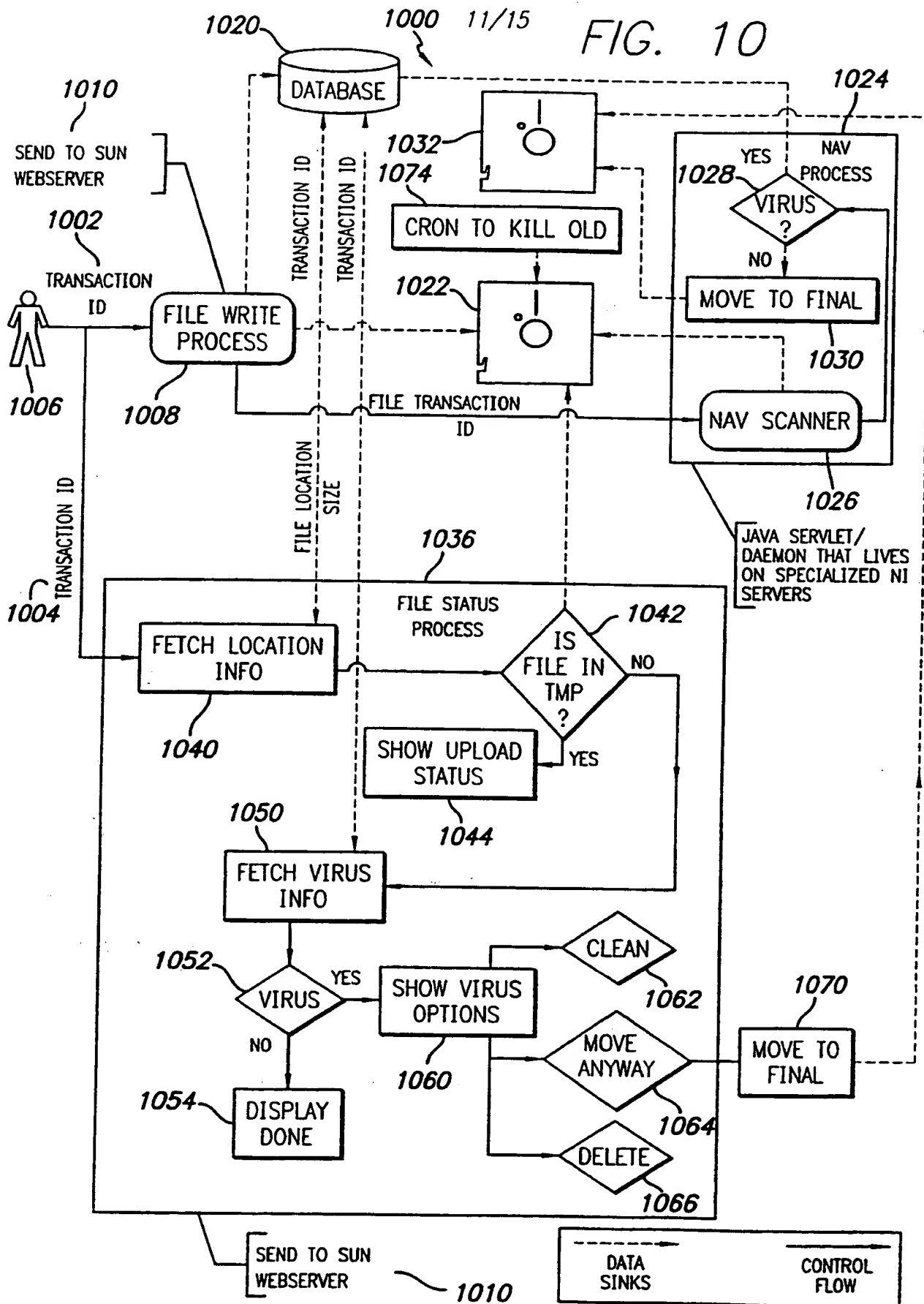
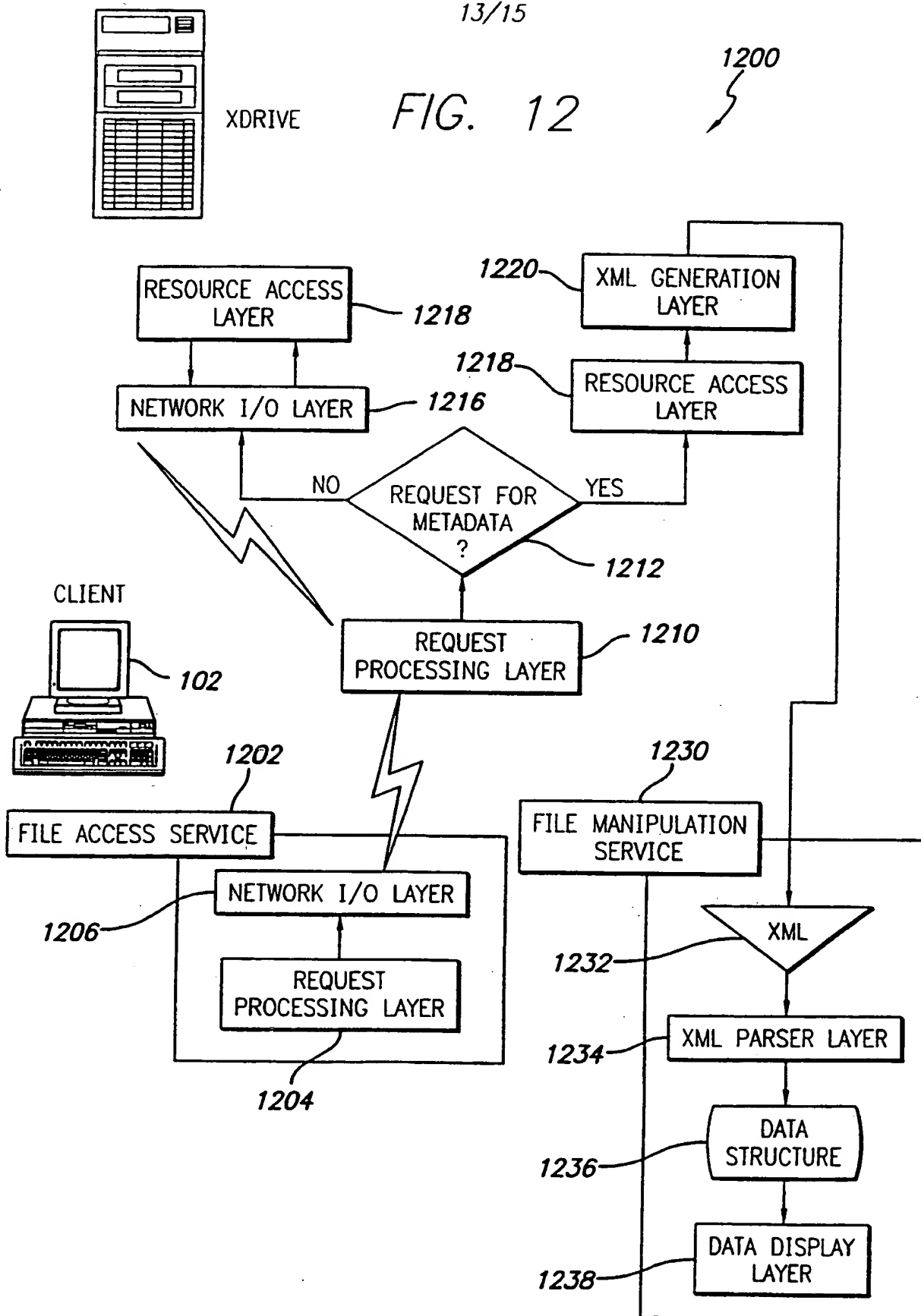


FIG. 9

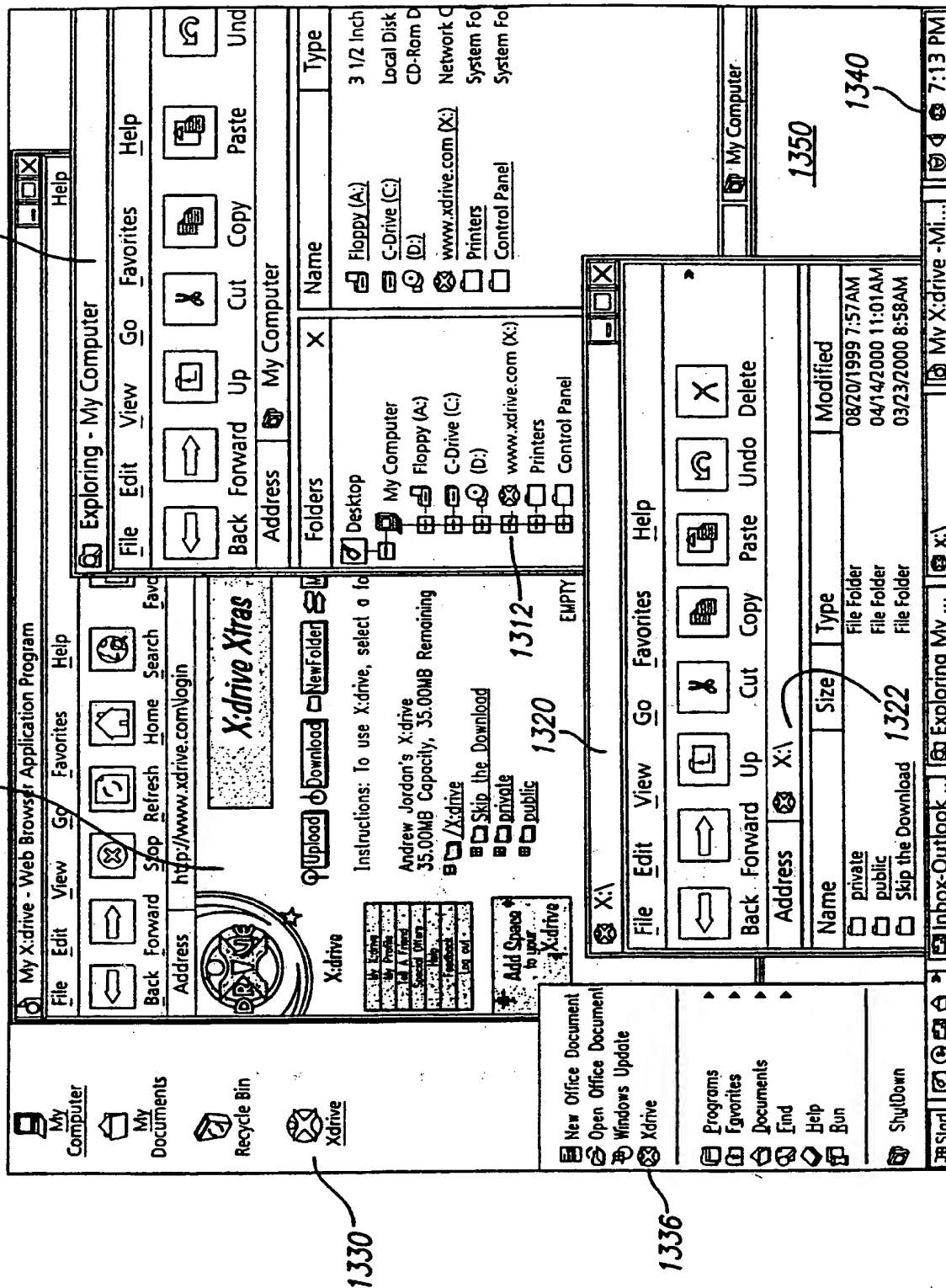


13/15



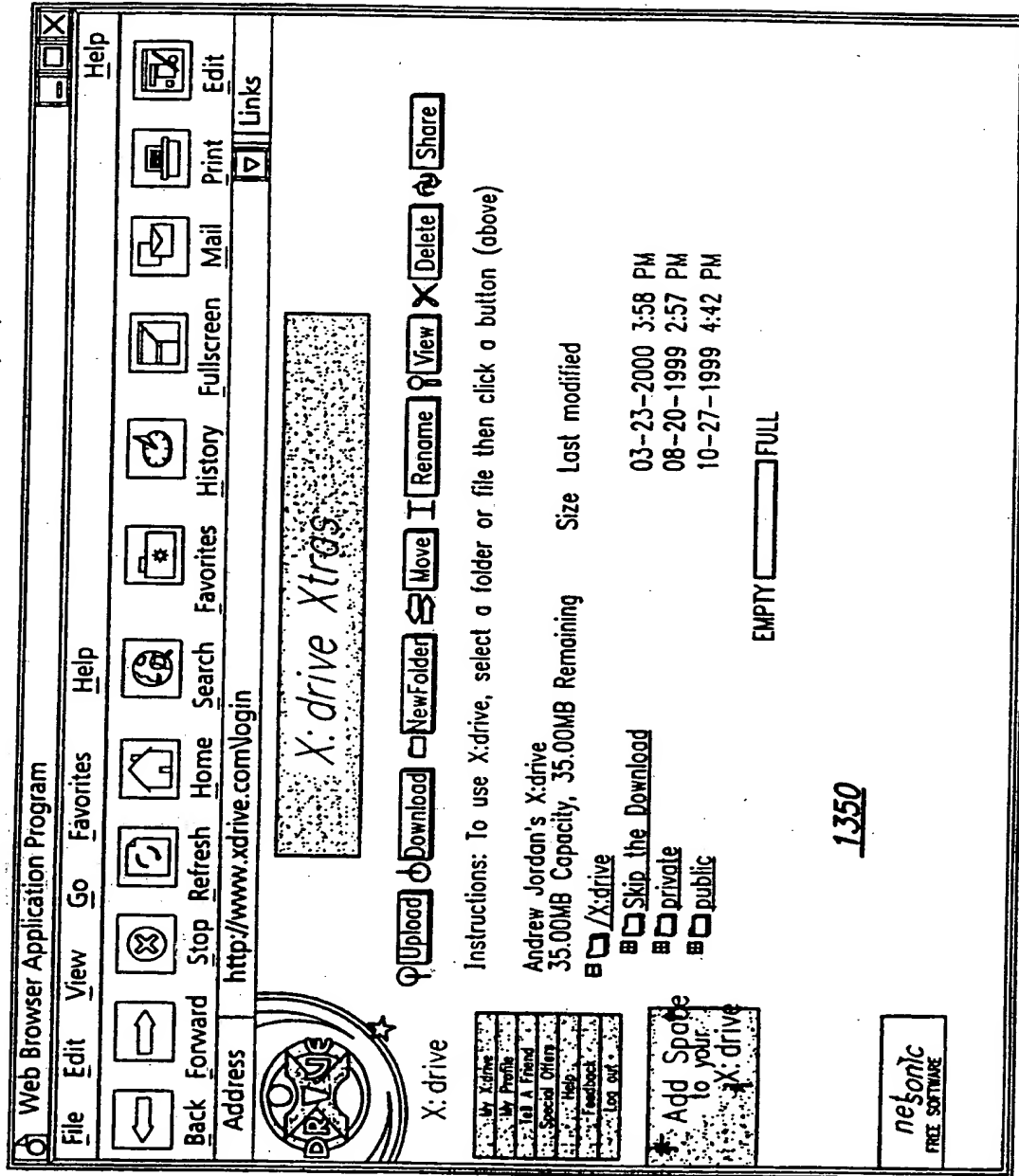
14/15

1350 FIG. 13



15/15

FIG. 14



SUBSTITUTE SHEET (RULE 26)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/30536**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(7) :G06F 15/00, 15/16, 17/30; B41B 15/00

US CL :345/326; 707/1,10; 709/104,105,212,213,217,226,229,245

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 345/326; 707/1,10; 709/104,105,212,213,217,226,229,245

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

East, West, IEEE

search terms : network, internet, storage, resource, parse, proxy

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,861,883 A (CUOMO et al.) 19 January 1999, col. 2 line 11 - col 6 line 37	1-50
Y	US 5,956,490 A (BUCHHOLTZ et al.) 21 September 1999, col. 2-6	1-50
Y,P	US 6,049,877 A (WHITE) 11 April 2000, col. 2 line 36 to col. 9 line 57	1-50
Y,E	US 6,154,738 A (CALL) 28 November 2000, col. 4 line 1 to col. 33 line 35	1-50
Y,E	US 6,151,601 A (PAPIERNIAK et al.) 21 November 2000, col. 8 line 35 to col. 25 line 67	1-50
Y,P	US 6,128,624 A (PAPIERNIAK et al.) 03 October 2000, col. 8 line 14 to col. 25 line 27.	1-50

☐ Further documents are listed in the continuation of Box C.☐ See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier document published on or after the international filing date	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search

10 JANUARY 2001

Date of mailing of the international search report

26 FEB 2001

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

FRANTZ B. JEAN

Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)*

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau(43) International Publication Date
10 May 2001 (10.05.2001)

PCT

(10) International Publication Number
WO 01/33381 A1(51) International Patent Classification?: G06F 15/00,
15/16, 17/30, B41B 15/00

(21) International Application Number: PCT/US00/30536

(22) International Filing Date:
3 November 2000 (03.11.2000)

(25) Filing Language: English

(26) Publication Language: English

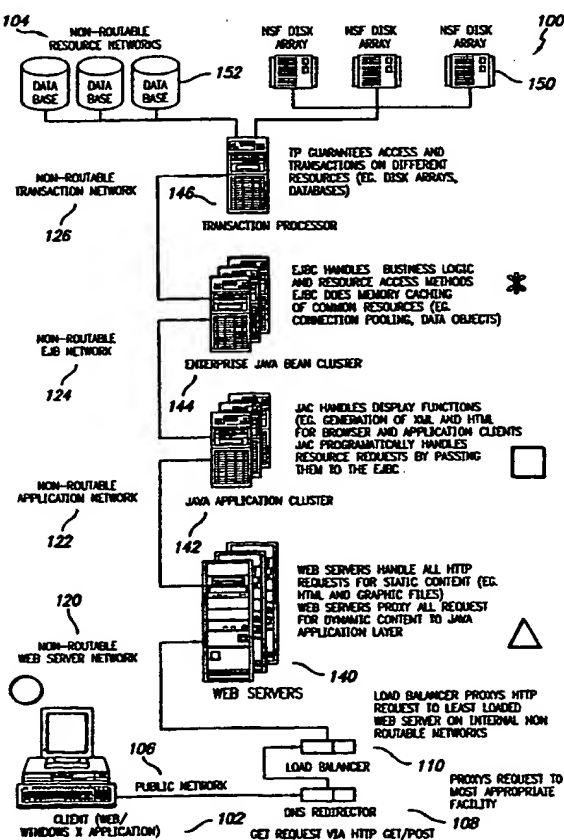
(30) Priority Data:
60/163,626 4 November 1999 (04.11.1999) US
09/570,583 12 May 2000 (12.05.2000) US(71) Applicant (for all designated States except US): XDRIVE,
INC [US/US]; Suite 2000 North, 1601 Cloverfield Boul-
vard, Santa Monica, CA 90404 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): O'BRIEN, Brett
[US/US]; XDrive, Inc., Suite 2000 North, 1601 Cloverfield
Boulevard, Santa Monica, CA 90404 (US). WHITELEY,
Sean [US/US]; XDrive, Inc., Suite 2000 North, 1601
Cloverfield Boulevard, Santa Monica, CA 90404 (US).
MCGREGOR, Lucas [US/US]; XDrive, Inc., Suite 2000
North, 1601 Cloverfield Boulevard, Santa Monica, CA
90404 (US). HALD, Martin [DK/US]; 111 Amherst
Aisle, Irvine, CA 92612 (US).(74) Agents: JORDAN, Andrew et al.; Cislo & Thomas LLP,
Suite 900, 233 Wilshire Boulevard, Santa Monica, CA
90401-1211 (US).(81) Designated States (national): AE, AG, AL, AM, AT, AU,
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ,
DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR,

[Continued on next page]

(54) Title: SHARED INTERNET STORAGE RESOURCE, USER INTERFACE SYSTEM, AND METHOD



(57) Abstract: The shared internet storage resources provides Internet-based file storage, retrieval, access, control, and manipulation for user. Additionally, an easy-to-use user interface is provided both for a browser or stand-alone application. The entire method provides means by which users can establish, use, and maintain files on the internet in a manner remote from their local computers yet in a manner that is similar to the file manipulation used on their local computers. A high capacity or other storage system is attached to the internet via an optional internet network that also serves to generate and direct metadata regarding the stored files. A web server (140) using a CGI, Java-based, or other interface transmits and retrieves TCP/IP packets or other internet information through a load balancer/firewall (110) by using XML to wrap the data packets. File instructions may be transmitted over the Internet to the Shared Resource System. The user's account may be password protected so that only the user may access his or her files. On the user's side, a stand-alone client application (142) or JavaScript object interpreted through a browser provide two means by which the XML or other markup language data stream may be received and put to use by the user. Internet-to-internet file transfer may be effected by directly downloading to the user's account space.

WO 01/33381 A1



HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

Published:

— with international search report

(48) Date of publication of this corrected version:

10 May 2002

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

(15) Information about Correction:

see PCT Gazette No. 19/2002 of 10 May 2002, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SHARED INTERNET STORAGE RESOURCE, USER INTERFACE SYSTEM, AND METHOD

TECHNICAL FIELD

This invention relates to resources on computer networks, particularly the Internet, and more particularly to a file storage and retrieval system that is available worldwide via the Internet which additionally allows a direct transfer of Internet files to an Internet storage, retrieval, and sharing resource. The present invention acts in the manner of a "Internet hard disk" or "Internet hard drive" to provide online storage and retrieval resources for users.

BACKGROUND ART

The Internet is the worldwide computer network making available a vast number of computer and information resources to institutions and individuals. A significant part of the Internet is the worldwide web that allows for web pages to be written in HTML and transmitted upon demand throughout the Internet. Recent developments have better established the use of XML (Extensible Markup Language) as a subset of SGML (Standard Generalized Markup Language, ISO standard 8879:1986). FTP (File Transfer Protocol) provides means by which files may be transferred over the Internet. All of these protocols are generally well known in the art, and collateral resources can easily be obtained to describe these further.

Patents relevant to the present invention include U.S. Patent No. 5,771,354 issued to Christopher Crawford on June 23, 1998 for an Internet Online Backup System, and U.S. Patent No. 5,901,228 issued to Christopher Crawford on May 4, 1999 for a Commercial Online Backup Service.

Additionally, portable programming systems such as Java®, JavaBeans, and JavaScript have been extensively developed with an anticipation of future portability across the vast network that is the Internet. Java®-related systems allow for object-oriented programming whereby objects or "beans" allow the passing of self-contained modules with associated processing methods that are used to act upon the accompanying data. Consequently, the "bean" can travel through a network and, under appropriate circumstances, have certain processes activated allowing manipulation of the information contained in the bean.

Advancements in Java®-related systems have given rise to the Enterprise JavaBean™ (EJB). The Enterprise JavaBean™ allows for clustering of servers such that the bean is given independence from specific servers on the system, yet can be activated or "instantiated" such that error recovery is easier, the system as a whole is more robust, and processing of the bean can be performed asynchronously so that all events do not have to happen at a pre-set time or serially/one after the other.

Enterprise JavaBeans™/EJBs allow serialization of beans. Such serialization allows the bean to be represented as a data stream of determined length. In essence, this is just a data file that is interpreted in the proper context, much the same as any electronic information file. Such serialization of the EJB allows it to be replicated and stored in case of catastrophic failure of a preferred server or the like.

If the server upon which the instantiated EJB dies, goes down, or fails, a previously replicated twin can be used to continue the process and allow for error recovery. More information about Enterprise JavaBeans™ technology can be found in the white paper, "Enterprise JavaBeans™ Technology: Server Component Model for the Java™ Platform" by Anne Thomas, revised December 1998, prepared for Sun Microsystems, Inc. and published/made available by the Patricia Seybold Group of Boston, Massachusetts.

Due to the nature of new technologies, terms such as "bean" or "instantiated" may seem unfamiliar to those new

to the pertinent art. Reasons for this include the difficulty of communicating quickly new and complex subjects as well as the good-humored nature of those who intensely pursue the establishment of new technology, particularly software systems. Consequently, for Java®-related systems, a coffee theme is often present that indicates to those knowledgeable in the art the general subject matter of interest. While distinctions may be subtle in the art, they can be very important and serve the ends of those attempting to establish, share, and forward the technology.

Generally, home pages or other web pages are requested by the user through designation of the URL (Uniform Resource Locator). With the transmission to the user via TCP/IP protocol, the information present at the URL (and generally a file located somewhere on a computer) is transmitted to the user. The file may have links, or pointers, to other resources including images, graphics, audio or video streams, or other resources. Mark-up language is used on the Internet in an attempt to provide an open-ended structure by which information of any sort that can be stored electronically (or perhaps even otherwise) can be made available to an end user on demand. As such, the Internet is seen as a powerful tool making almost any information resource available to any computer or to any person using a computer.

Over the past several years, the personal computer has increased in power and capacity as commercial demand has driven the research and development of producers and vendors. It is now not uncommon to be able to easily find an Intel-manufactured 500 megahertz Pentium®-based system having well over 10 gigabytes of hard disk space, as well as 32 - 256 megabytes of RAM. As such, the power by which files may be received and acted upon by the local user through his or her PC has kept pace with the advances in technology.

However, there currently remain obstacles to universal access to an individual's own information stored on his or her computer. First of all, computers are very heavy. They are bulky. They generally weigh several kilograms and are not easily transportable. Lightweight laptop computers or the like generally do not have the same resources available to the user as a regular PC. Additionally, access to local area networks (LANs) is generally not available once the computer leaves the premises occupied by the LAN. Additionally, Internet access is often restricted by the use of a modem. Modems generally provide data transmission speeds on the order of 56 kilobits per second. This is approximately the same as 7 kilobytes per second. However, headers and other information are required to properly transmit information over the Internet and increase the effective size of files.

Even with the increased availability of broad band access to the Internet, it becomes an important feature of electronic information processing and the like in order to provide resident resources on the Internet. Such resources could include the sharing of files and the like in a manner that are easy to use and understand.

Due to these and other restrictions regarding data transport, transmission, and reception, a need has arisen for means by which files and other data may be available worldwide through the Internet and not tied to a local computer. The present invention addresses this demand by providing means by which files and other data may be stored on the Internet and made available worldwide through the Internet.

DISCLOSURE OF INVENTION

The present invention provides an "Internet hard drive" or "Internet hard disk" to and from which files may be stored and retrieved. Denominated commercially as "X:Drive," the present invention allows users to store files of foreseeably any type on a resource available throughout the Internet. Once available to the Internet, the files stored on the user's X:Drive are available to the same extent as the Internet, namely worldwide.

Note should be made that the term "X:Drive" refers both to the system as a whole and to the individual space allocated to an individual user. Consequently, reference is sometimes made herein to the X:Drive system or to X:Drive to refer to the system as a whole. At other times, the term X:Drive indicates the user's individual X:Drive, or allocated

space. The different uses are indicated by context.

In order to effect the Shared Internet Storage Resource of the present invention, a central or distributed storage facility is provided. First and foremost is the high-speed access storage facility where files are actually stored. Such individual storage areas may be allocated in individual limited allotments, or be left open-ended and limited only by the capacity of the physical devices responsible for storage. Metadata, that is data about the files stored on the network hard drives or other storage devices, is generated and stored in a separate database. The database of metadata (the metadatabase) and the network-attached storage facility may be linked by an internal network. It is possible for the database to be stored on the same network storage facility or device on which user files are also stored. System management may select whether or not to distribute or consolidate the database with the network storage.

Also attached to the internal network is a web server that serves to generate and transmit the information to the Internet, and ultimately the user. The web server files may pass through a load balancer and/or firewall before proceeding on to the Internet. The same is similarly true for information coming into the web server from the Internet.

XML may be used in combination with JavaScript or the like to provide two means by which the Shared Internet Storage Resource of the present invention may be achieved. The first is a JavaScript object which may be transmitted to a browser program running on the user's computer. Such browsers may include ones that are well known, including Netscape® Communicator and Microsoft® Internet Explorer. Alternatively, a stand-alone application may be installed and stored upon the user's computer. This stand-alone application serves to intermediate the user commands with the web server and ultimately the metadatabase in the Internet storage device.

As an additional enhancement, the user interface may be a client program that meshes seamlessly with standard user presentations in WYSIWYG (what you see is what you get) graphic user interfaces (GUIs). As such, a drive may be shown on the user's computer and may be denominated "x:" (or "y:" or "z:", etc., depending upon user preferences). The user can then read from or write to the x:\ Shared Internet Storage Resource drive much in the same way as you would the local a:\ and c:\ hard drive.

When the user shuts down his or her computer, information that is stored on the Shared Internet Storage Resource of the present invention remains on the Internet. The user can then access such information from another computer, another geographic location, or even give permission to share files on the Shared Internet Storage Resource with others. Password protection or other security protocols may be used to limit or discriminate access to the user's files.

The Shared Internet Storage Resource of the present invention allows for direct Internet-to-Internet file transfer to a user's allocated X:Drive file space in a process referred to as "Skip the Download" or "Save to My Xdrive."

BRIEF DESCRIPTION OF DRAWINGS

Figure 1 is a schematic view of the X:Drive system of the present invention. The different tier levels are shown, along with the marking indicia of a circle, triangle, square, and star/asterisk corresponding to the same indicia in Figure 3.

Figure 2 is a schematic view of Java® library objects operating in the transactions or data exchanges occurring in the present invention.

Figure 3 is a detailed flow diagram showing the operation of the present invention. Indicia including a circle, a triangle, a square, and a star/asterisk correspond to tier levels shown in Figure 1 and indicate the level of operation of the steps shown in the flowchart of Figure 3.

Figure 4 is a flowchart showing the operation of the XDFile Enterprise JavaBean™ (EJB) used in the present invention.

Figure 5 is an overview of the Java® architecture used to effect transactions in the present invention.

Figure 6 is an alternative schematic diagram of the Java® architecture shown in Figure 5.

Figure 7 is a schematic and flowchart diagram showing the IO (input/output) for the database transactions of the present invention.

Figure 8 is a schematic diagram of the data recovery process as effected by the FileIO component of the XDFile object used in the present invention.

Figure 9 is a schematic depiction of failure recovery elements.

Figure 10 is a schematic and flowchart diagram showing virus protection effected in the present invention.

Figure 11 is a schematic and flowchart diagram of the Internet-to-resource transfer ("Skip the Download"/"Save to My Xdrive") as set forth in the present invention.

Figure 12 is a schematic and flowchart diagram of the client system used in the present invention.

Figure 13 is a Windows™ desktop display showing both the client and web-browser applications.

Figure 14 is a display of a web browser pointing to a user's X:Drive.

BRIEF DESCRIPTION OF APPENDICES

Appendix 1 is a listing of web site/server code use to achieve the present invention.

Appendix 2 is a listing of the code used on the client side to achieve the present invention in a Microsoft® Windows™ environment.

Appendix 3 is a listing of the JavaScript code used to achieve the present invention in a Sun Microsystems® Java® environment (including one on a browser).

MODE(S) FOR CARRYING OUT THE INVENTION

The detailed description set forth below in connection with the appended drawings is intended as a description of presently-preferred embodiments of the invention and is not intended to represent the only forms in which the present invention may be constructed and/or utilized. The description sets forth the functions and the sequence of steps for constructing and operating the invention in connection with the illustrated embodiments. However, it is to be understood that the same or equivalent functions and sequences may be accomplished by different embodiments that are also intended to be encompassed within the spirit and scope of the invention.

Appendices 1, 2, and 3 provide the source code for, respectively, the Web Site/Server Code of the X:Drive Shared Internet Storage Resource system of the present invention; the Windows Client Code; and the JavaScript Listings for the present invention. These Appendices are incorporated herein by this reference thereto as if set out in their entirety. It is contemplated that these Appendices provide a full, complete, and enabling disclosure to those of ordinary skill in the art or less by which the present invention may be achieved.

Additionally, the reference numbers used in conjunction with the figures are numbered such that the 100's place of the number indicates the number of the drawing figure. For example, the 600 series of reference numbers refers to Figure 6, while the 200 series refers to elements shown in Figure 2.

The present invention provides a method by which an Internet hard disk or hard drive may be achieved in a manner similar to a hard disk or hard drive available locally to the individual on the local computer. Additionally, as Internet use becomes a more familiar and everyday event for people, the resources provided by the present invention may allow the actual use of the Internet hard drive or X:Drive set forth herein to act as such a resource with the files being called up for execution for programs available and processed either locally and/or over the Internet. In light of the

foregoing, it can be seen that the present invention may act as a bridge or may pave the way towards a more inter-networked community for the use and processing of electronic information.

The virtual disk drive provided by the present invention may be selectively shared with others or kept entirely private. Additionally, and as set forth in more detail below, the use of a metadatabase provides quicker access and the ability to distribute the information regarding the legion of X:Drive accounts over a wide geographic area, enabling redundant preservation of user information by server clusters implementing Enterprise JavaBeans® (EJBs), or otherwise.

The Shared Internet Storage Resource, User Interface System, and Method set forth herein is generally referred to as "X:Drive." Context reveals whether or not the term X:Drive is referring either to the system as a whole or the individual's own account.

The X:Drive system of the present invention uses network application practices and may rely upon Java® Enterprise JavaBeans™ (EJBs) to enable distributed and clustered computing and file management environment. Along with such Java®-based and network-oriented design, the X:Drive system of the present invention also contemplates the use of open programming standards such as XML and Web-DAV (Web-based Distributed Authoring and Versioning). The use of such technology is foreseen as providing wide support by the user community as well as speed and development, refinement, and polishing.

As shown in Figure 1, the X:Drive system 100 has a multi-tiered, network-based application infrastructure. The multi-tiered nature of the system allows it to separate operations in an efficient manner. The network-based aspects of the X:Drive system allows it to disperse resources geographically as well as allow a high degree of communication between different aspects or facets of the system.

The X:Drive system may be considered enabling technology as a medium that is independent of the applications and uses to which it is applied. The X:Drive system is currently based on object-oriented principles with each application layer responsible for a discreet functionality or aspect of operation. Both hardware and software resources may then successfully experience heavy re-use with both scalability and flexibility inherently provided. While these advantageous aspects of the X:Drive system are achieved, as a multi-tiered system, X:Drive involves a higher cost of complexity and planning. Thus, those who would seek to wrongly copy the X:Drive system would do so without accruing the great expense in time and money necessary to achieve the present X:Drive system. They would ride on the backs of those who not only developed the system, but also those who got it to work right and in a commercially-reliable manner.

The use of tiers in the X:Drive system of the present invention is realized in both the network systems and the application systems involved in achieving X:Drive.

As shown in Figure 1, a variety of tiers, or layers, are present between the client 102 and the ultimate data resources 104. Between the client 102 and the data resources 104, are one or more layers or tiers, accomplishing the following.

The client 102 may be coupled to a public network 106 (such as the Internet) that may include a DNS redirector 108 as well as a load balancer 110. The public network 106 may then lead into a web server network 120. The web server may then lead into an application network 122, which in turn leads into an EJB (Enterprise JavaBeans™) network 124. The EJB network 124 may lead into a transaction network 126, which in turn leads into the data resources 104.

The client 102 may be either a web- or browser-based application or an application resident on a Windows™ X system (the X indicating the version of Windows applicable, i.e., Windows® 95, Windows® 98, Windows® 2000, etc.). Requests generally originate from the client as the X:Drive system 100 is one that operates at the command of users directing the client program. Client requests may be made versus the Hypertext Transfer Protocol (HTTP) GET/POST

function. In a preferred embodiment, the GET/POST operation may be augmented with Web-DAV extensions to the HTTP protocol. Commands are transmitted by the client 102 are sent to the DNS redirector 108, which then isolate the request via a proxy server process. A proxy server process prevents a direct connection between the client 102 and the other downstream resources in the X:Drive system 100. Such proxy serving prevents inadvertent or mischievous disruption of service by allowing only certain commands or information to be propagated through the X:Drive system 100. This prevents mischievous users from disrupting the system as such rogue commands are intercepted by the proxy server and denied further propagation.

After the client command has passed through the DNS redirector/proxy server 108, the request by the client 102 is then directed to the most appropriate facility. As the X:Drive system is scalable, facilities may be distributed geographically, even over the face of the globe. This allows, at the outset, more efficiencies to take place in the X:Drive system 100 of the present invention so that more users may be served more quickly and so that the advantageous features of the X:Drive system may be realized by the widest number of users in the quickest way possible.

Due to the construction and architecture of the X:Drive system 100, a number of machines/servers running a number of different processes may be distributed over a wide area. Broad band or high-speed access as provided by Internet backbone or the like may allow the X:Drive system to be effectively carried out over the entire face of the planet. The scalability and flexibility of the present invention augments its utility. Such advantages are further advanced by efficient use of the resources so that greater and better service can be provided.

Upon receiving the request from the client 102, the DNS redirector 108 transmits the requests on to a load balancer which may provide a second proxy process under HTTP protocol and transmit the request to the least-loaded and most-available web server on an internal, non-routable, or other server network 120.

The web server network 120 may be non-routable and may comprise a number of individual machines or servers processing the HTTP or other requests from one or more load balancers 110. Each of the web servers 140 in the network 120 may handle HTTP requests for static content, such as HTML and graphic files. The web servers may proxy all requests for dynamic content to a Java® application network 122.

As used in the X:Drive system 100 of the present invention, the Java® application networks may be non-routable. The use of non-routable facilities in the X:Drive system 100 of the present invention indicates their operation in a local area network (LAN). However, between tiers, the individual networks themselves may be available such that a web server 140 in Illinois may pass requests for dynamic content to Java® application clusters 122 in Wisconsin.

Each Java® application cluster 122 may be composed of a number of Java® application servers 142 with each server 142 handling display functions necessary for user accounts, including the generation of XML, HTML, and other instructing displays for either browser or application clients 102. If a Java® application cluster 122 receives a resource request from the web server tier 120, the Java® application cluster 122 will pass the resource request onto the Enterprise JavaBean™ EJB network tier 124.

As for the web server 120 and Java® application networks 122, the EJB network 124 may also be non-routable and operate upon a LAN. The EJB network may be an EJB cluster having a number of EJB servers 144. Each EJB cluster handles the business logic and resource access methods and protocols required for the resource requests and management. The EJB cluster (EJBC) caches memory of common resources such as the pooling of data connections and the like, as well as data objects. Resource access requests and transmissions are then passed out to the transaction network tier 126, which may also be non-routable. The transaction network tier 126 has a transaction processor 146 which controls, operates, and guarantees access and transactions on different resources. These different resources are the ultimate data resources 104 that may include NFS (Network File Server) disk arrays 150 and databases 152. The NFS

disk arrays 150 may supply the actual storage capacity for the files of generally any size. The databases 152 comprise records of information regarding each of the files (metadata) stored by the NFS disk arrays 150 under the X:Drive system 100.

By bifurcating the file information in databases 152 separate from the actual files themselves on the NFS disk arrays 150, file information and user queries can be handled much more quickly as display components of the present invention are important to provide the user information regarding the status and availability of the files stored on the X:Drive system 100. Consequently, although a user may have a hundred separate files in an X:Drive directory, he or she may be only interested in one. Consequently in order to provide the user the information necessary to make the decision as to which file to receive, move, rename, delete, or store, the use of the database provides a very quick and easy means by which such user requests can be satisfied. It is anticipated that the actual use of the file storage facilities on the NFS disk arrays 150 or the like may comprise only a part of the operations of the present invention. Having the ability to display, select, and determine file operations is one of the useful advantages provided by the X:Drive system 100 of the present invention.

Note should be taken of the non-numerical indicia present in Figure 1. Most notably, a circle is associated with the client 102, a triangle with the Java® application cluster 122, a square with the EJB network 124, and a star/asterisk with the transaction network. These non-numerical indicia correspond to those set forth in Figure 3. As different actions are performed at different tiers in the present invention, the non-numerical indicia provide an easy or visual means by which the operation of the different tiers can be indicated in Figure 3.

Figure 2 shows a logic diagram in sequence structure for the Java® library objects used in the X:Drive system 100 of the present invention. Generally, throughout the description of the X:Drive system 100 of the present invention, the prefix XD indicates "X:Drive." For example, in Figure 2 the steps/status indicators of XDError stands for X:Drive Error, and XDXML stands for X:Drive Extensible Markup Language. Likewise, the use of the term XDFile indicates X:Drive File as a Java® library object effecting and intermediating the file operations of the present invention.

In Figure 2, the Java® system 200 allows operations to be performed on the metadatabase 202 and the operating system (OS) File System 204. Additionally, the XDFile object 210 may activate or instantiate the Database.Search object 216. The XDFile object 210 may be activated, or invoked, by the FileAction object 220. The FileAction object 220 may also activate the Database.Search 216 and Database.BigSearch 222 objects. Operations of the Java® library objects in the system 200 as shown in Figure 2 may be contingent upon the SessionSecurity object 224, which may instantiate or use the Database.Search object 216 and/or the Database.Transaction object 214. The SessionSecurity object 224 may return a separate object 226 to the UserData object 230. The Database object 236 may inherit or transmit from its Transaction 214, Search 216, and/or BigSearch 222 objects.

The information generated may then be transmitted to the Database 202 for meta-information and the OS File System 204 for the actual data. If an error is generated during the operation of the Java® library object system 200, an XDError object 240 may serve to handle the error while a successful operation may be returned in the form of the XDXML object 242. In the Java® library object system 200 of Figure 2, the Database 202 may contain intelligence or programming for connection to SQL databases and the like. Options regarding the operations of the database 202 may be read from a configuration file. The Database object 236 may be able to connect multiple databases for redundancy in the case of repeated or redundantly archived information, or for functionality in order to connect to that database which responds most quickly to the requests and commands.

The Database object 236 determines which database operation to perform and/or to which database to send operations based on the type of request it receives. For example, transaction requests may demand a separate database

from those of regular query and BigSearch 222 requests. In order to maintain more efficient operation, the Database object 236 generally sends session users to the same database whenever possible so that latency and database replication is not passed on to the user.

The Database.Transaction object 214 is able to handle larger SQL statements such as those that would cause a load on the database. The Database.Transaction object 214 may spawn children classes that handle the transaction logic in order for more efficient operation.

The Database.Search object 216 is designed to handle smaller SQL statements and has children classes for specific search types, such as those along anticipated and common fields or types of information.

The Database.BigSearch object 222 handles larger, non-transactional SQL statements such as those used for reports in system accounting, monitoring, or otherwise. Children classes of the Database.BigSearch object 222 would handle specific large searches such as those that might be implemented on a monthly or other periodic basis.

The FileIO object 212 inherits and overrides Java®'s data file object. The file object contains logic to engage multiple disks or resources for redundancy and/or functionality and contains the functionalities necessary to manipulate files on the OS File System 204. The FileIO object 212 may react to the JMS (Java Messaging Service) events triggered by events on the disks of the OS File System 204.

Alternatively, one or more monitoring objects may be used to gather pertinent status information regarding the OS File System 204. When monitoring objects are used, the FileIO objects then query the common monitoring objects to determine the state of the system. In the present system, the monitoring object is denominated the Mount Point Status bean, or MPS bean, 534 (Figures 5 and 9).

Additionally, disk level transactions are carried out by the FileIO object 212. Under the management of the FileIO object 212, user accounts are able to span or traverse several disks. The spanning of such several disks enables better recovery from failure should an error occur or system resources become unavailable in an unpredictable manner. The XDFile object 210 uses FileIO 212 to handle the file system transactions. By using the Database.Transaction file object, the XDFile object 210 handles database file transactions. The XDFile object 210 coordinates transactions for both the FileIO object 212 and the Database.Transaction file object 214 to keep both synchronized and to handle failure should it occur.

The UserData object 230 holds user data for a session of the X:Drive system. A session is basically a span of time for which a user engages the X:Drive system. Methods are included in the UserData object 230 to manipulate the user status, so that the activity may be monitored, as well as whether or not the user has logged in.

The SessionSecurity object 224 uses web logic session mechanisms to create the UserData object 230. It does this by returning a separate object 226. The SessionSecurity object 224 authenticates a user's login and expires old sessions with re-direction of such old sessions to appropriate pages.

The FileAction object 220 may have children classes and contain logic for determining request types such as user requests, administration requests, etc. Tests for file action requests such as quotas and permissions, etc., may also be handled by the FileAction object 220. The FileAction object 220 accesses the file methods in the XDFile object 210.

The XDError object 240 reads a configuration file of error lists which gives each error an I.D. number. Such error lists preferably pivot on the language in which the X:Drive system 100 of the present invention is programmed. Such lists should also be able to pivot on the partner with which the X:Drive system 100 operates. Default values for the lists may be to X:Drive errors in the English language. The XDError object 240 preferably holds errors in a stack and returns any such errors from the stack. Additionally, the XDError object 240 preferably accepts new errors by code or by message.

The XDXML object 242 accepts an object and delivers as output an XML representation of a transaction or status requested by the user or client software.

Figure 3 shows the data flow through the X:Drive system 100 of the present invention, particularly that as reflected by the tiered configuration shown in Figure 1. From a starting point 300, a request is sent by HTTP POST/GET command at step 302. Web-DAV protocol may also be used and is currently considered preferable. The send request is implemented on the client 102 and is evaluated by the web server 120 as a request for static content in step 304. If the request is for static content, the file is served by the web server 120 at step 306, and the file is displayed at step 308 by the client 102.

If at step 304 the request for static content is evaluated as negative, a proxy request is issued by the web server network 120 to the Java® application cluster 122 at step 312. The request is received by the Java® application cluster (JAC) 122 and submitted to a servlet at step 314. The Java® application cluster (JAC) 122 then parses the request header at step 316. The Enterprise JavaBean™ (EJB) network 124 then authenticates the request at step 318. If authentication cannot be achieved, process control is then re-directed to the re-login page via the JAC network 122 at step 320. If authentication succeeds at step 318, the JAC network 122 then parses the multi-part form data at step 324.

The JAC network 122 then determines the type of request at step 326. The request is then submitted to the FileAction EJB 220 at step 328. The EJB network 124 then evaluates the request at step 330 in order to ensure that all the business rules and other applicable limitations are met, such as quota limitations, permissions, and the like. If the evaluation is successful at step 330, the EJB network 124 then submits the request to the XDFile EJB 210 at step 332 and on to the transaction processor 146. The appropriate actions are then taken via the transactional database 152 and the disk arrays 150. If the business rule evaluation 330 fails, an error may be generated and, as for other errors in the data flow process of Figure 3, a session error object 334 may be generated in a session error stack 336.

In effecting the data transfer to the ultimate system resources 104, evaluation is made as to the operation in step 340. If the operation is not a data read operation such as a directory listing or file read, the error stack is checked at step 342. If an error has occurred, the error status is sent to the client 102 at step 344. The client 102 then accepts the transmitted XML code and renders the appropriate display for the user at step 346. If the error stack evaluation step 342 does not reveal any error, a success message is generated at step 350, and the subsequently-generated XML is received by the client 102 and displayed by the user at step 346.

If at the evaluation step 340, the operation is not a data read action, the error stack is checked at step 352 much in the same way as it was at step 342. If an error has occurred, the error status is sent to the client 102 at step 354. The error status message is then received as XML code by the client 102 at step 346 and displayed to the user. If at evaluation step 352 the error stack reveals no errors, the evaluation is then made by the EJB cluster as to whether or not the operation is a file read at step 360. If the operation is a file read, the data stream is converted to a network stream and transmitted as a file to the client 102 by the Java® application network 122 at step 362. The data is then accepted by the client 102 and served to the user at step 364.

If at evaluation step 360 the operation is not a file read (see Figure 4), then by elimination, the action is a request for file metadata such as a directory listing indication of file attributes or the like. At step 366, the metadata retrieved from the database 152 is then translated into XML format by the EJB cluster 124. The XML data is then transmitted to the JAC network 122, which encapsulates the XML from the network and sends it on to the client at step 368. The JAC network 122 then sends the encapsulated XML to the client 102 for rendering and display at step 346.

As indicated in the description above with regards to Figure 3, users utilizing the client system 102 to connect to the X:Drive system 100 do so via the public Internet and then submit requests and receive replies effecting or indicating

the user's requests. Requests for file manipulations, such as uploads, downloads, copies, moves and updates travel through each functional layer of the X:Drive system 100.

The core of the EJB cluster, and as indicated in Figure 2, the XDFile EJB provides core effectiveness in the present X:Drive system 100. The XDFile EJB 210 is a multi-tiered component. The X:Drive system 100 stores file metadata (such as directory structure, file name, file attributes, etc.) in the database 152 for fast retrieval, sorting, searching, linking, and other capabilities beyond standard file systems. The actual file data is stored by the X:Drive system 100 in network-attached storage units or storage area networks such as those shown in Figure 1, the NFS disk arrays 150.

To access files that exist in this hybrid environment (bifurcated between file information and file data), X:Drive uses the XDFile object 210 to manipulate both files and file data in two-phase committal transactions. Figure 4 shows the details of these transactions.

In Figure 4, the XDFile EJB system 400 allows entry at any one of the five darkened triangles. If the action is to be a copy, entry is made at the copy entry point 402. If the action is a file read, entry is made at the file read point 404. If the action is a file write, entry is made at the file write point 406. If the action is a file delete, entry is made at the delete point 408. If the action is a file move, entry into the XDFile EJB 210 is at the move entry point 410.

Beginning first with a file copy action beginning at the copy point 402, the evaluation of the operation occurs at step 420, where determination is made whether or not the action is a read transaction. If the action is a read transaction, program flow proceeds onto the read action and entry point 404. The corresponding database action 424 is then taken. As the action is a read transaction, the corresponding database record is read and evaluation is made as to whether or not the database action, in this case read action, has been successful at step 428. If the read action is not successful, the changes are then rolled back, if any, at step 432. An error is then returned at step 436 and the XDFile object awaits further instructions. If the evaluation at step 428 regarding the database action was successful, action can then be taken on the actual file itself on the OS File System 204 at step 440. In the present case, the FileOS Action 440 is a read action, and the file may be read into a temporary buffer or other memory space. The FileOS Action is evaluated for success at step 444. If the FileOS Action step 440 was unsuccessful, a fatal error is returned at step 448, and the changes, if any, are rolled back at step 452. If the evaluation at step 444 was successful, evaluation is made as to whether or not the action was a copy read at step 456. If the action was a copy read, return is made to the copy entry point 402 at step 464 in order to perform the write portion of the copy function. If the evaluation at step 456 indicates that the action was not a copy read action, evaluation is made at step 468 to determine if the action was a move/copy action. If the action was a move/copy action, control is then directed towards the move entry point 410 via step 472 in order to delete the original file as the success of the move/copy transaction at evaluation step 444 indicates the success of the file write step of the FileOS Action step 440. Program control is then turned over to the move/action entry point 410 so that the original file may be deleted at its original location via the delete entry point 408.

If the move/copy evaluation step 468 indicates that not only was the action not a copy read, it was also not a move/copy, then the action is committed to the system at the ultimate system resource level 104 at step 480 and an indication of success is then returned at step 484.

Upon reaching the move entry point at 410, evaluation is made at step 490 to determine whether or not the transaction is a copy transaction. If it is a copy transaction, the program then enters and executes the copy entry point 402. If not, the delete entry point 408 is activated to effect the remainder of the move transaction.

Consequently, it can be seen that a variety of actions take place depending upon the state of the XDFile EJB 210 at the database action 424 and FileOS action 440 steps.

In performing file reads and writes, simple one-step actions are taken because neither of these read or write actions are either copy reads 456 or move/copy 468 and so they fall into the system commit 480 and return a successful indication at step 484. The same is generally true for the one-step delete action. Consequently, whenever a user wants to read, write or delete a file, entry can be made into the respective entry points at 404, 406, and 408. Errors are returned when necessary.

However, the copy action 402 and the move action 410 require multiple loops through the XDFFile EJB 210 in order to effect their operations. For the copy function 402, the initial read must be made successfully with the evaluation step 456 then prompting the write step to occur by the return to the copy entry point at step 464. The read transaction step 420 is then evaluated in the negative and the write entry point/action 406 is invoked with the database action occurring at step 424 to write the new information to the transactional database 152 and, if successful, the FileOS write action for the data at step 440. If the file write is successful, the evaluation at step 456 as to whether or not the action is a copy read is answered in the negative as is the evaluation of the transaction as to whether or not is a copy transaction executed under the move action at step 468. The resources are then committed, temporary resources are released, and the success indication is returned at step 484.

Consequently, for a copy transaction 402, the loop is first made through the read function 404 and then the write function 406. For the move action at entry point 410, a copy transaction is first executed with the two-loop operation as set forth previously. Upon completion of the copy action, the delete action 408 is implemented in order to erase the original file and its file data. Upon the third loop through the delete step 408, the transaction is neither a read under the copy command at step 456 nor a copy under the move command at step 468. Consequently, the move function has successfully completed, the system resources are committed at step 480, and a success indicator is returned at step 484.

In Figure 5, an overview of the Java® architecture of the X:Drive system 100 of the present invention is shown. The Java® architecture 500 shown in Figure 5 may generally arise from the client 102. A file action container 504 has certain attributes and operations as do the other beans of the architecture 500. Contained within the file action container 504 are a number of stateful, stateless, and entity beans, as well as other containers having other beans. The file action container 504 contains two stateful beans: a user data stateful bean 506 and a process request stateful bean 508. The user data stateful bean 506 has a user info entity bean 510 and a security stateless bean 512.

The process request stateful bean 508 contains a single container, the XDFFile container 520. The XDFFile container 520 contains three (3) beans and a container. The three beans of the XDFFile container 520 are: a database IO stateful bean 522, a file IO stateful bean 524, and an admin stateful bean 526. The container is a recovery container 530 which contains a recovery IO stateful bean 532, a mount status stateful bean 534, a recovery admin stateful bean 536, and a recovery process stateful bean 538.

As indicated by the nature of the beans carried by the containers, stateful beans generally carry information about the state of the bean, process, or otherwise as useful information for the ends and operations of the X:Drive system 100 of the present invention. Stateless beans generally carry no state information, and entity beans are generally for information or identification only. As Java® beans are objects intended to carry both data and processes in association with one another, it is up to the operations of the X:Drive system 100 of the present invention to selectively and appropriately activate the beans and enable the proper actions to take place. The file action container 504 is shown in alternative representation in Figure 6. In Figure 6, a client 102 issues a user authentication request 602 and an operation request 604. The user authentication request 602 is passed into the user data stateful bean 506 in the file action container 504. The operation request 604 is passed into the process request stateful bean 508. The user information entity bean 510 then transmits information to a user information database 610, as does the security stateless bean 512. The process

request stateful bean uses a first property file 612 that is loaded upon deployment of the XDFFile container 520. The property file is loaded into the admin stateful bean 526 for use with the OS file system 204. A Java® transaction server 620 may operate in conjunction with the database 152 as well as the OS file system 204 in order to process the operation request 604. The second property file 630 may be loaded by the recovery admin stateful bean 536 upon the bean's deployment. The recovery IO stateful bean 532 and the recovery admin stateful bean 536 both transmit information to the recovery queue storage buffer 640. The mount status bean 534 operates in conjunction with the mount status of the system 650.

The recovery container 530 is called when once a failed resource begins to recover. Further description of the recovery process is given below. However, Figures 5 and 6 operate in tandem to show linearly (Figure 5) and organically (Figure 6) the structure and operation of the XDFFile object 210.

Figure 7 shows the detail of the XDFFile database component. A transaction processor (such as Tuxedo from BEA) works in conjunction with the database transaction object 214 as well as the FileIO object 212 to provide a robust and reliable system. Both the database transaction 214 and the FileIO 212 objects include logic and/or programming to handle situations where database or disk array access cannot be guaranteed. The database.transaction object 214 handles the inherent doubt present in the system by using replicated or repeated clusters of databases. The replication process creates latency or delay, in the system. In order to accommodate this latency, the database transaction object 214 uses a session object (a data construct representing a user session on the X:Drive system 100) to determine if the user's request can be transferred, or replicated, from one database cluster to another, in case of future system failure.

An important aspect with respect to the reliable operation of the X:Drive system 100 is the need to separate databases into functional groups. While the query database may be optimized for quick and small queries and while a transaction database might be optimized for fewer, larger, more time consuming updates, the database layer 236 in the X:Drive system 100 allows for associating SQL commands with different database clusters based on functionality. Additionally, the X:Drive database layer 236 is configured for consolidation and addition of databases on the fly.

As shown in Figure 7, the SQL command 710 is issued and passed to a SQL command evaluator 712. A SQL evaluator determines the SQL type so that the SQL can be sent to the appropriate database type (that is, in the X:Drive system 100, the transaction database 150, the query database 152, or both).

Upon determining the database type of the SQL statement 712, the database preference is evaluated at step 714 to determine if the user should be sent back to the same database. If the user is not to be sent back to the same database, the database currently bearing the least load is found in step 716, and query is then made in step 718 to ensure that the selected least-loaded database is still up, running, and available. If it is, a specification regarding the pooling of database resources is created 720 and transmitted to the database object 236. Database object 236 then takes the SQL command and passes it to the appropriate database, either the transaction database 150 or the query database 152 via associated connecting pools 730.

If at step 718 the least loaded database is not available, an alternative database must be used and query is made at step 736 to determine whether or not the alternate database is up. If the alternate database is not up and the evaluation step 736 fails, additional databases may be queried or, as indicated in Figure 7, a fatal error may be generated at step 738. If the alternate database is up, a pool specification 720 is generated and passed to the database object so that the SQL command may be implemented upon the transactional 152 databases via the connection pools 730.

If at step 714 the user must be sent back to the same database, query is made at step 740 to determine if that database is still up. If it is, the request is passed to the pool specification 720 where it is subsequently passed to the database object 236; on to the connection pool 730, and the appropriate database, either the transaction database 150 or

the query database 152. If the same database is not up and the evaluation at step 740 fails, an alternative database must be used, but the SQL request is queried at step 744 to determine if the SQL command is transferable to the alternate database. If not, a fatal error occurs at step 746. If the SQL command is transferable, query is made at step 750 to see if the alternate database is up and active. Should the evaluation fail, subsequent databases may also be queried if the SQL command is transferable. However, as shown in Figure 7, if the second database is unavailable, a fatal error may be generated at 746. Otherwise, the database is up, and the evaluation at step at 750 is successful and the command is made available to the database object 236 via the pool specification standard 720 and on to the databases through the connection pools 730.

In order to ensure proper operation of the XDFile database object 210, a database status monitor 760 persistently and on-goingly queries the databases 150, 152. The status is then returned to a database status object 762. the database status object may provide information to the recovery container 530 of the XDFile object 210.

The recovery mechanism for the X:Drive system 100 of the present invention is shown in Figure 8. The FileIO object 212 uses a recovery object such as the recovery container 530 to handle write transactions 406 (as opposed to read transactions 404) when the transaction processor 214 fails. The recovery object is transparent to the user, making it easier and more convenient for the user to use the X:Drive system 100 while decreasing the concern that such a user would have in case of a power outage or other failure in one part of the X:Drive system 100.

The FileIO object 212 reports an error to the user, but informs the user that her request was stored in the X:Drive system 100 and that the X:Drive system 100 will try to apply the change as soon as possible. If the storage unit, represented as a mounting point in the EJB cluster becomes unavailable for write transactions 406, the monitoring client 760 updates the EJB network 124 that the status of the mounting point is "down." Once the mounting point is available and checked for data integrity, the status is updated from "down" to "recovery" and the recovery object 530 is called to apply all queued requests for the file action container 504. This keeps the user from catastrophically losing uploads and other file writes, but may cause some delay in file reads.

In the recovery system 800 of the present invention, the multi-connected pooled database object, the recovery-enabled FileIO object 212, and the transaction processor 146 work together to create a resource layer offering high availability, recovery, and scalability. Additionally, this resource layer (encapsulated in the XDFile EJB 210) lends itself to replication of the data, both geographically and locally. Such replication preferably has the three essential traits of being off-site, application-driven, and accessible. With this level of controlled replication, secondary X:Drive clusters are enabled in geographically diverse locations in order to enhance the reliability of the X:Drive system 100. Consequently, data loss from one data center or even the physical loss of an entire data center would not cause loss of customer data or access. Re-direction would occur dynamically and this information would be replicated in a plurality of sites across the X:Drive system 100, the query or metadata databases provide multiple pointers to the user's data.

In the recovery system 800 of Figure 8, the recovery system is initially initiated when the MPS Bean 534 is set for a mode to detect mount point recovery at step 804. At step 804, a recover method is called and the external mount point is checked. Query is made at step 806 to evaluate whether or not recovery is already occurring. If recovery is already occurring, an exception is thrown at step 808 and exit is made at this finish point. If recovery is not already occurring, a list of mount points in recovery mode is generated in step 810. Additionally, at step 812 a list of mount points which are down is also generated. Query is made at the evaluation step 818 as to the presence of available recovery objects in the recovery queue. If no such objects are available in the queue, the disk or other database is set into the "up" mode at step 820. The queue for that disk is then unlocked in step 822, and the recovery process is complete at step 824. If at evaluation step 818 recovery objects are still in the queue, evaluation is made as to whether or not the system has gone

past the lock count at step 830. If so, the queue for the disk in recovery is locked at step 832 for both the lock count evaluation 830 and the queue lock 832 step, control is then directed to the evaluation step as to whether or not the target file exists 834. If the target file does not exist and the evaluation at step 834 fails, the recovery object is removed from the queue at step 840. The status of the recovery is subsequently put in the request for alert queue at step 842 and return is then made to the query step 818 to determine whether or not objects are still available for recovery in the queue.

If the target file does exist when evaluated at step 834, evaluation is made as to whether or not the request is more current than the file at step 850. If the request is older than the current file, the recovery object is removed from the queue at step 840, and the status for the request is put in the request or alert queue 842 and control returns back to the evaluation step 818 to see if any further recovery objects are available in the recovery queue.

If, in evaluating the request, it is found that the request is more current than the file, the request is submitted to the XDFFile object 210 at step 852. The submission of the request to the XDFFile object 210 is not recoverable. If the submitted request is successful as indicated by the evaluation at step 854, the recovery object is removed from the queue at step 840, its status is put into the request for alert queue at step 842 and evaluation is made at step 818 as to the presence of any additional recovery objects in the recovery queue. However, if in submitting the request to the XDFFile object 210 at step 852 the submission fails, query is made at step 860 as to whether or not the mount point has gone down. If at step 860 the mount point is still up, the request from this mount point is ignored at step 862 and the queue for the disk is unlocked at step 864. Control of the program is then returned to the recovery object availability query in evaluation step 818.

As shown in Figure 9, the mount point status bean 534 has UP, DOWN, and RECOVERY states. This bean is applicable to the file database 150, as well as user disks 970, 972 as well as recovery disks 974, 976. Additionally, the recovery admin stateful bean 536 is directed towards the recovery database 980 in order to effect the recovery process 800.

In order to effect virus scanning and repair features, the X:Drive system 100 preferably uses the Java® JNI (Java Native Interface) to access a Norton Anti-Virus or other dynamically linked library (NAV.DLL) to scan files for viruses via a Java® servlet. The Java® servlet runs on a Windows™ version X server and can use JNI to make calls to the NAV.DLL dynamically linked libraries. In effect, the Windows™ X machine becomes a specialized NAV.DLL server located at the EJB network layer 124 of the X:Drive system 100, on a sub-network of the resource network. The logic integrating the NAV.DLL dynamic linked libraries with all X:Drive file writes is shown schematically in the flow diagram in Figure 10.

As shown in Figure 10, the virus scanning sub-system 1000 takes the file/transaction ID 1002 and a transaction ID 1004 from a user 1006. The file/transaction ID 1002 is passed to a file write process 1008 executed by a SUN® or other web server 1010. The file is written to both the database generically indicated at reference 1020 and to a temporary file storage area 1022. The file write process 1008 passes the file transaction ID to the Norton Anti-Virus (NAV) process 1024. Within the NAV process 1024 is NAV scanner 1026. The NAV scanner monitors the data stream or otherwise to determine and detect the presence of any viruses. If upon evaluation the NAV process 1024 detects a virus at evaluation step 1028, data sink action is taken with respect to the database 1020. If no virus is detected, the sequence moves to its final termination at step 1030 and data sink action is taken with respect to a temporary file on medium 1032.

While both the file and transaction ID 1002 are delivered to the file write process 1008, the transaction ID alone 1004 is transmitted to a fetch location info step 1040 on a SUN® or other web server 1010. The fetch location info step 1040 transmits its results to an evaluation step 1042, which determines whether or not the file is in the temporary storage area 1022. If the file is in the temporary area, the file's upload status is shown in step 1044. If the file is not in the

temporary medium 1022, virus information is fetched at step 1050 in the file status process 1036.

Once the virus information has been fetched, it is evaluated as to whether or not there is a virus present at step 1052. If there is no virus detected, then the virus evaluation terminates and a display of same may be made at step 1054.

However, if evaluation step 1052 indicates the presence of one or more viruses, a plurality of virus options may be shown and presented to the user at step 1060. Among the virus options available are: the cleaning of the virus at step 1062, moving the virus to a different location at step 1064, and/or deleting the virus in step 1066. If step 1064 is taken with the move of the virus-laden file despite its infectious nature is made, movement of the file with its final destination is made in step 1070.

As shown in Figure 10, a number of data sink actions are taken with respect to information. Additionally, as indicated by Figure 10, the NAV process 1024 is a separate entity and may be considered to be a JAVA® servlet/daemon living on specialized Windows® NT or other servers.

In order to make resources available on an on-going basis to the virus scanning sub-system 1000 of the present invention, a chron file 1074 (a file executing commands on a periodic basis according to the time) is used to remove old files from a first temporary storage resource 1002.

Figure 11 shows the Skip the Download/Save to My Xdrive system where a file on the Internet can be transferred over to an individual's X:Drive at generally data speeds far faster than those available to the end user. This allows the user to exercise dominion and control over the file without having to bear the burden of downloading it to the local computer at the present moment. Once the transfer has taken place across the Internet from the host to the X:Drive system 100, then the user may download the file stored in his X:Drive directory to his local computer at his convenience.

As X:Drive exists on the Internet network, transferring a file from one network resource (such as a web or FTP server) to the user's X:Drive is made much faster from the user's standpoint by by-passing the local connection to the user and allowing the user to submit the transfer request directly to the X:Drive network for execution. The X:Drive system 100 then downloads the requested data from the target server to the user's X:Drive over the presumably higher speed connections of the public Internet.

As shown in Figure 11, the Save to My Xdrive system 1100 first has the user 1110 submit the URL at step 1112. In order to access the X:Drive system 100 of the present invention, the user submits the URL as well as his or her user name and password at step 1114. Upon submitting the URL and the appropriate verification information, evaluation is made of the information for authentication purposes at step 1116. If the evaluation fails and authentication is not achieved, a login form is displayed in conjunction with the previously-indicated URL at step 1118. If the request is authenticated, it is submitted to the STD/STMX (Skip the Download/Save to My Xdrive) queue 1132 at step 1130. A status process is then spawned at step 1134.

Save to My Xdrive status is then checked on an on-going basis by using the queue in the temporary storage area at step 1136. Query is made as to whether or not the transfer is complete at step 1140. If the transfer is complete at step 1140, then the successful completion is indicated to the user at step 1142. However, if the transfer is not complete, query is made as to the presence of any transfer errors at step 1146. If an error has occurred, an error message is displayed to the user at step 1148. However, if the transfer is incomplete but no errors have occurred, the same is then displayed to the user at step 1150, and a short pause is taken at step 1152 for re-invoking the check STD process at step 1136.

Once the STD queue 1132 receives the request, a daemon process processes the request from the STD queue at step 1160. Query is made as to the business logic of the queued request at step 1162. If the request fails the business logic check 1162, the status is updated at step 1164. Control may transfer back to the STD queue 1132.

If the business logic check succeeds at step 1162, the URL site is contacted by the X:Drive system 100 at step

1170 and the download process is activated. The data transmitted by the URL is then saved in temporary X:Drive space in step 1172, with the data being transferred then to the user data space at step 1174. The URL site 1180 may exist anywhere on the Internet so long as it is available to the X:Drive system 100. In a similar manner, a temporary storage space 1182 may also exist anywhere on the Internet so long as it is accessible and controllable by the X:Drive system 100.

Upon transferring data to the user's data space as shown in step 1174, query is made as to the success of the transfer at step 1188. For either success or failure of the successful file transfer at evaluation step 1188, the status is updated at step 1164 and is passed on to the STD queue 1132 until either success or an error is finally achieved. The status process spawned at step 1130 monitors the update status generated by step 1164 and displays the status to the user during and after the download of the file from the Internet to the user's X:Drive system.

Figure 12 shows a schematic and flowchart diagram for the client system generally used under Microsoft® Windows™ for achieving the present invention. The X:Drive system offers its clients two basic services: a file access service by which files can be uploaded and downloaded to and from X:Drive, as well as a file manipulation service from which file metadata can be obtained and manipulated. Both of these services rely upon the context of their usage. For example, the web client of the present invention uses native upload and download features as well as dialogs in the user's web browsers to facilitate the service.

With the use of the web browsers on the local machine, Windows® X clients use the Windows™ TCP/IP stacks inherently present with the Windows® version X operating system. All the file transfers effected by the X:Drive system can take place as HTTP POST/GET or, preferably, Web-DAV transfers. Generally, two basic layers are present in the file manipulation servers of the X:Drive system 100 of the present invention. An XML parser operates in conjunction with an XML data displayer. By coordinating the two basic layers of the file manipulation service, the server is able to respond with generally the same XML code to all clients. The client is then responsible for converting the XML to a relevant data structure and displaying the XML in an appropriate context. In the present invention, the JavaScript web client receives the XML code and parses it into a JavaScript data structure. A display layer in association with the client and/or browser renders the data structure as an HTML document. The Windows® X client parses the same XML code, but the display layer renders the data structure into a device listing that is understood by the Windows® version X operating system. The importance of this layered architecture is that it generally makes trivial the creation of new clients. Instead of simply creating dynamic web pages (and thus limiting service to web browsers alone), the X:Drive system 100 can enable many platforms, such as operating systems, without altering the server structure. Most platforms come with some sort of XML parsing layers, and many platforms come with display layers ready made. Consequently, the time to market may generally be considered low and efficient establishment and implementation of the X:Drive system 100 of the present invention can be achieved fairly quickly. Additionally, expansion into new platforms generally becomes much quicker as no alteration of the server structure generally needs to occur as Java® and related program functionalities are highly portable from one system to another.

In the client system 1200, as shown in Figure 12, the client 102 has a file access service 1202, including a request processing layer 1204 coupled to a network I/O layer 1206. Commands and data are then transmitted to the server side of the X:Drive system 100 where the server side request processing layer 1210 transmits the data to a query evaluating whether or not the request is one for metadata at step 1212. If the evaluation fails and the request is not one for metadata, the network I/O layer 1216 and the resource access layer 1218 are invoked in order to provide access to and operation of the transaction database 152.

If the request for metadata query at step 1212 succeeds, the request is passed on to the resource access layer 1218

and on to the XML generation layer 1220. The response to the request from the metadatabase 150 is transmitted to the file manipulation service system 1230 of the client 120. The XML transmitted by the XML generation layer 1220 is received by the file manipulation service 1230 as well as its XML handler 1232. The XML is then passed on to the XML parser layer at step 1234 to arrive at a data structure 1236 that is then ready for display and so is passed on to the data display layer 1238 for display to the user who may then re-initiate the process by implementing the file access service 1202.

Figure 13 shows the X:Drive system 100 as implemented on a Windows™ X machine, in this case, a Windows '98 machine (an Intel-based personal computer running the Microsoft Windows '98 operating system).

The second frontmost window 1310 of Figure 13 is headed by the inscription "My Computer" and shows the presence of a drive at logical letter X: 1312 with the X:Drive logo and the label www.xdrive.com (X:). This is an example of the user interface provided by the client application. The X:Drive system is transparent to the user and functions as any other drive present on the system.

If the user were to click on or activate the X:\ drive on the My Computer window 1310, the second window 1320 appears (partially obscuring the "My Computer" window 1310) and shows the listing under the X:\ Drive. The address of the window 1320 shows the location of the directory as being at X:\ 1322.

Also shown in Figure 13 is the desktop icon 1330, the start menu icon 1336, and the system tray icon 1340. These icons accompany the client program 102 and provide greater functionality for the user. Each icon serves to activate the client program in accordance with user-settable preferences.

Figure 13 also shows the web-based application 1350 in the background, behind the My Computer 1310 and X:\ 1320 windows. The web-based application window 1350 is shown in Figure 14. Note should be taken of the exact correspondence between the directory structures of web-based application window 1350 and the client-based application window 1320. This correspondence provides the user with a uniform, familiar, and dependable interface upon which the user can rely.

As set forth above, the three accompanying Appendices are incorporated herein in their entirety, as is the previously filed provisional application.

While the present invention has been described with regards to particular embodiments, it is recognized that additional variations of the present invention may be devised without departing from the inventive concept.

INDUSTRIAL APPLICABILITY

It is an object of the present invention to provide a Shared Internet Storage Resource on which users may store and retrieve files to make them available to themselves, or possibly others, throughout the Internet.

It is an additional object of the present invention to provide all manner of file access and control generally available to files local to the users for such Internet-stored files.

It is an additional object of the present invention to provide an easy-to-use and readily understood user interface through which files may be stored, retrieved, and manipulated on the Internet.

It is an additional object of the present invention to gather metadata regarding such files and to store such metadata in a database.

It is yet another object of the present invention to provide a plurality of means by which Internet-stored files may be manipulated and controlled.

It is yet another object of the present invention to provide a browser-based access to Internet-stored files.

It is yet another object of the present invention to provide stand-alone application access to Internet-stored files.

It is yet another object of the present invention to provide means by which Internet files may be stored on an Internet resource by a direct Internet-to-Internet transfer subject to the control of a remote or limited-resource user.

These and other objects, advantages, and the industrial utility of the present invention will be apparent from a review of the accompanying specification and drawings.

Web Site/Server Code

###addspace.cgi.....	1
###client_info.cgi.....	9
###cookie.cgi.....	12
###download_client.cgi.....	19
###email_change.cgi.....	21
###error.cgi.....	23
###explorer.cgi.....	24
###explorer_user_data.cgi.....	28
###file_load.cgi.....	30
###file_save.cgi.....	34
#! file_upload_stat.cgi.....	36
###folder_create.cgi.....	42
###forgot_password.cgi.....	44
###forgot_username.cgi.....	47
###frame_generic.cgi.....	50
###get_a_shared_file.cgi.....	52
###get_a_shared_file_download.cgi.....	54
###login.cgi.....	57
###logout.cgi.....	65
###navbar.cgi.....	66
###password_change.cgi.....	68
###promo.cgi.....	71
###removespace.cgi.....	74
###selected_delete.cgi.....	79
###selected_rename.cgi.....	80
###settings_save.cgi.....	82
###share_a_file.cgi.....	85
###signup_account.cgi.....	91
###signup_form.cgi.....	105
###signup_success.cgi.....	111
###signup_toc.cgi.....	113
###skip_the_download.cgi.....	115
###skip_the_download_status.cgi.....	122
###tell_a_friend.cgi.....	131
###web_unauthorized.cgi.....	135

###addspace.cgi

```

#!/usr/bin/perl
#####3
##  addspace.cgi - processes additional space requests using Epoch's
##  do_approval library function
##  written by Karen Eppinger
#####3

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Error;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::Deal;
use XDrive::DatabaseO::Table::Item;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::Client::Actions;
use XDrive::Client::Quota;
use XDrive::Sale::Purchase;
use Mail::Sendmail;
use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Template;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::CGI::Cookie;
use EpochClient_ssl;

use strict;

$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)};    # Make %ENV safer

&main();

#####
## main: main function calls all others
##
##
#####

sub main
{
    ##the hash that will be filled in and send to the Epoch function
    my %hData;

    my $oCGI = CGI->new();
    my $oErrors = new XDrive::Error;
    my $oDBH = XDrive::DatabaseO->new();

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    ####
    ## Validate the user and if an error happens during
    ## the validation process die redirect to the error cgi
    ####
    my $oToken = xd_security_check($oDBH,$oCGI,$oErrors);

    if ($oErrors->Occurud)

```

```

    {
        xd_fatal_error($oCGI,$oErrors);
    }

    $hData{'ipaddr'} = $oCGI->remote_addr();

    if ($hData{'ipaddr'}=~/^192.168.2/)
    {
        $hData{'ipaddr'}='0.0.0.0';
    }

    my $sUserName = $oToken->data('user');
    # my $sPartnerCode = $oToken->data('partner_code');
    my $sPartnerCode = $oCookie->getElement('partner');
    my $oTemplate = new XDrive::Template
    (
        {
            'partner_code' => $sPartnerCode
        }
    );

    ##used to figure whether to give user the form or process the form
    my $sAction = $oCGI->param("action");

    ## if the action is a request type, we give the user the form
    if ($sAction eq 'process')
    {
        ##get the date from the form already pre-screened by javascript
        my $returnValue = GetFormData(\%hData,$sUserName,$oCGI,$oDBH);
        if ($returnValue)
        {
            ##call the Epoch function that processes the transaction
            my $sReturnCode = do_approval(%hData);

            ##if we've been approved $return will contain a number that
            is
            ##7 characters and starts with a Y followed by 7 digits
            ##only change user's quota if approved
            ##else let them know there was a problem; all problems
            start with N
            ##return code could be logged in our database to track
            tranactions

            ##truncate expressions longer than 32 characters
            if (length($sReturnCode)>32)
            {
                $sReturnCode = substr($sReturnCode,0,32);
            }

            if ($sReturnCode=~m/^Y/)
            {
                ##if transaction went through, give them more space
                ##and show them the ok screen
                my $error =
                &WriteToPurchaseDatabase($sReturnCode,\%hData,$sUserName,$oDBH);
                if ($error)
                {
                    &TransactionOK($sReturnCode,
                    \%hData,$sUserName,$oTemplate,$oDBH,$oToken,$oCGI,$oErrors);
                    $oDBH->commit();
                }
                else
                {
                    ##error inserting into the database
                    &TransactionBad('141',$oTemplate,$oErrors);
                }
            }
        }
    }

```

```

        $oDBH->rollback();
    }
    elsif ($sReturnCode=~m/^N/)
    {
        ##tell them there was a problem
        ##for some reason we get this returned with
        $sReturnCode=~s/~//;
        my $error =
&WriteToFailedDatabase($sReturnCode, \%hData, $sUserName, $oDBH);
        &TransactionBad($sReturnCode, $oTemplate, $oErrors);
        $oDBH->commit();
    }
    else
    {
        ##There was a problem connecting to server
        my $error =
&WriteToFailedDatabase('COULDNOTCONNECT\n', \%hData, $sUserName, $oDBH);

        &TransactionBad('COULDNOTCONNECT\n', $oTemplate, $oErrors);
        $oDBH->commit();
    }
}
else
{
    ##this is someone trying to use the
    ##bogus card numbers and isn't one of us
    ##don't bother writing to database because
    ##it is caught before going to Epoch
    &TransactionBad('NMYBADCARD\n', $oTemplate, $oErrors);
}
$oDBH->disconnect();
}
elsif ($sAction eq 'intro')
{
    &ShowIntroPage($oTemplate, $sPartnerCode, $sUserName, $oToken, $oCGI, $oErrors);
}
else
{
    &ShowForm($oTemplate, $sUserName, $oErrors);
}
exit;
}

#####
## GetFormData:  Fills in the hash that is required by Epoch's function
## Fill in one field at a time because not all fields on the page should go
## into hash plus a few fields don't come from form
#####

sub GetFormData(\%, $, $, $)
{
    my $hData = shift;
    my $sUserName = shift;
    my $oCGI = shift;
    my $oDBH = shift;

    my $value = 1;

    ##these are mandatory to process the transaction
    ##javascript checks insure users fill the fields with the proper data

```

```

$hData->{'transtype'}='approve';
$hData->{'co_code'}='xdr';
$hData->{'pi_code'}= $oCGI->param("pi_code");
$hData->{'cardnum'}= $oCGI->param("cardnum");
$hData->{'cardexp'}=$oCGI->param("cardexp");

##someone is trying to access from a site other than ours and use the
free credit card
if (($hData->{'cardnum'} eq '4121371122223333') || ($hData->{'cardnum'}
eq '4111111111111114'))
{
    if ($hData->{'ipaddr'} ne '0.0.0.0')
    {
        $value=0;
    }
}

##not required but used to check for fraud
$hData->{'cardname'}= $oCGI->param("cardname");
$hData->{'street'}=$oCGI->param("address");
$hData->{'city'}=$oCGI->param("city");
$hData->{'state'}=$oCGI->param("state");
$hData->{'zip'}=$oCGI->param("zip");
$hData->{'phone'}=$oCGI->param("phone");

##get email out of the database
my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
$oDiskAccount->loadWhere('USERNAME',$sUserName);
my $sUserSeq = $oDiskAccount->fetchColumn('USER_SEQ');
$oDiskAccount->finish();

my $oUserInfo = XDrive::DatabaseO::Table::UserData->new(undef,$oDBH);
$oUserInfo->loadWhere('SEQ',$sUserSeq);
$hData->{'email'}=$oUserInfo->fetchColumn('EMAIL_ADDRESS');
$oUserInfo->finish();

return $value;
}

#####
## ShowIntroPage: called to show the intro page
##
#####

sub ShowIntroPage($,$,$)
{
    my $oTemplate = shift;
    my $sPartnerCode = shift;
    my $sUserName = shift;
    my $oToken = shift;
    my $oCGI = shift;
    my $oErr = shift;

    my ($nUserSeq, $oUserData);

    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);
    my $quotaAvailable = $oAction->QuotaFree();
    $quotaAvailable = sprintf("%2.2f",$quotaAvailable/1024);

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,undef);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

```

```

my $nUserSeq = $oDiskAccount->fetchColumn("USER_SEQ");

my $oSearch = XDrive::DatabaseO::Search->new($oDiskAccount-
>fetchDBO());
my $items = $oSearch->XDGetItemsForSale($nUserSeq);

my $itemString='';
my $i;
for $i(0..$#{ $items })
{
    ##now using the code, get the description for the item in the
    ##proper language. This is kept in List.pm
    my $code = "EPOCH_$items->[$i][1]";
    my $description = $oErr->ReturnMessageGivenCode($code);
    $itemString .= "<LI>$description";
}

## Load the required template HTML files.
$oTemplate->load('addspace_intro.shtml');
$oTemplate->tags
(
    'products' => $itemString,
    'quota' => $quotaAvailable
);
$oTemplate->clear;

print "Content-type: text/html\n\n";
print $oTemplate->get();
}

```

```

#####
## ShowForm: called to show the user the blank form
##
#####

```

```

sub ShowForm($,$)
{
    my $oTemplate = shift;
    my $sUserName = shift;
    my $oErr = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,undef);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);
    my $nUserSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oSearch = XDrive::DatabaseO::Search->new($oDiskAccount-
>fetchDBO());
    my $items = $oSearch->XDGetItemsForSale($nUserSeq);
    my $oDeal = XDrive::DatabaseO::Table::Deal->new(undef, $oDiskAccount-
>fetchDBO());

    my $itemString='';
    my $i;
    for $i(0..$#{ $items })
    {
        $oDeal->loadWhere("ITEM_SEQ", $items->[$i][0]);
        my $spi_code = $oDeal->fetchColumn("PRODUCT_CODE");
        my $code = "EPOCH_$items->[$i][1]";
        my $description = $oErr->ReturnMessageGivenCode($code);
    }
}

```

```

        if ($i == 0)
        {
            $itemString .= '<input type="radio" name="pi_code" value="" .
$pi_code . '" CHECKED>' . $description . '<BR>';
        }
        else
        {
            $itemString .= '<input type="radio" name="pi_code" value="" .
$pi_code . '">' . $description . '<BR>';
        }
    }

    $oDeal->disconnect();

    ## Load the required template HTML files.
    $oTemplate->load('addspace_request.thtml');
    $oTemplate->tags
    (
        'products' => $itemString
    );
    $oTemplate->clear;

    print "Content-type: text/html\n\n";
    print $oTemplate->get();
}

#####
## WriteToFailedDatabase:  if the transaction fails write it to the failed
## _transactions table
#####

sub WriteToFailedDatabase($,\%,$, $)
{
    my $sTransCode = shift;
    my $hDash = shift;
    my $sUserName = shift;
    my $oDBH = shift;

    my %transInfo;
    ##write transaction info into database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

    $transInfo{'user_seq'} = $oDiskAccount->fetchColumn('USER_SEQ');
    $oDiskAccount->finish();
    $transInfo{'trans_code'} = $sTransCode;
    $transInfo{'product_code'} = $hDash->{'pi_code'};
    $transInfo{'IP'} = $hDash->{'ipaddr'};

    my $intoDB = XDrive::Sale::Purchase->new($oDBH);
    my $error = $intoDB->FailedTransaction(\%transInfo);

    return $error;
}

#####
## WriteToPurchaseDatabase:  write the user transaction info to th
## user_purchase
## table
#####

sub WriteToPurchaseDatabase($,\%,$, $)

```

```

{
    my $sTransCode = shift;
    my $hDash = shift;
    my $sUserName = shift;
    my $oDBH = shift;

    my %transInfo;
    ##write transaction info into database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

    $transInfo{'user_seq'} = $oDiskAccount->fetchColumn('USER_SEQ');
    $transInfo{'account_seq'} = $oDiskAccount->fetchColumn('USER_SEQ');
    $oDiskAccount->finish();
    $transInfo{'trans_code'} = $sTransCode;
    $transInfo{'product_code'} = $hDash->{'pi_code'};

    my $intoDB = XDrive::Sale::Purchase->new($oDBH);
    my $error = $intoDB->Checkout(\%transInfo);
    return $error;
}

#####
## TransactionOK:  if the tranaction was processed and ok'ed, we add the
proper space to the
## user's xdrive and let them know the space has been added
#####

sub TransactionOK($,\%,$,$)
{
    my $sTransCode = shift;
    my $hDash = shift;
    my $sUserName = shift;
    my $oTemplate = shift;
    my $oDBH = shift;
    my $oToken = shift;
    my $oCGI = shift;
    my $oErr = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);
    my $userSeq= $oDiskAccount->fetchColumn('USER_SEQ');

    my @aCodes=split(/\|/, $sTransCode);
    $aCodes[1]=~s/~//;
    my $sNewQuota;
    my $sAddedSpace;

    my $oDeal = XDrive::DatabaseO::Table::Deal->new(undef,$oDBH);
    $oDeal->loadWhere('PRODUCT_CODE', $hDash->{'pi_code'});
    my $itemSeq = $oDeal->fetchColumn('ITEM_SEQ');
    my $oItem = XDrive::DatabaseO::Table::Item->new(undef,$oDeal-
>fetchDBO());
    $oItem->loadWhere('SEQ', $itemSeq);
    my $sCode = "EPOCH_" . $oItem->fetchColumn('CODE');
    my $sDescription = $oErr->ReturnMessageGivenCode($sCode);

    my $sSpaceToAdd = $oItem->fetchColumn('NAME');

    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);
    $sNewQuota = $sSpaceToAdd + $oAction->QuotaLimit();

```



```

##now set the new quota
##in the database and in the ncftpd database
##used during testing to reset occasionally
##$sNewQuota = 25600;
XDQuotaLimit($sUserName, $sNewQuota);

##insert into the spool to update ftp account

## Load the required template HTML files.
$soTemplate->load('addspace_ok.shtml');
$soTemplate->tags
(
  (
    'transactionCode' => $aCodes[1],
    'addedSpace' => $sDescription
  );
$soTemplate->clear;
print "Content-type: text/html\n\n";
print $soTemplate->get();
)

#####
## TransactionBad: If we get an error code beginning with and N, it's a
declined tranaction
## get the error code and give user the bad tranaction page with error code
#####

sub TransactionBad($,$)
{
  my $sTransCode = shift;
  my $soTemplate = shift;
  my $soErrors = shift;

  if ($sTransCode!~/^\d+$/ )
  {
    ##error codes contains
    $sTransCode="EPOCH_" . $sTransCode;
    chop($sTransCode);
  }

  ##$soErrors->AddErrorByErrorCode($sTransCode);

  $soErrors->AddErrorByCodeIncludes($sTransCode);
  my $sReturnError=$soErrors->Message($sTransCode);

  if(!$sReturnError)
  {
    $sReturnError = "The was an problem processing your transaction.
Please try again.";
  }

  ## Load the required template HTML files.
  $soTemplate->load('addspace_bad.shtml');
  $soTemplate->tags
  (
    (
      'error' => $sReturnError
    );
  );
  $soTemplate->clear;
  print "Content-type: text/html\n\n";
  print $soTemplate->get();
}

```

###client_info.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});
use CGI;

exit &main;

sub main ()
{
    my $oCGI = CGI->new();

    ##get this info from Michael Ryan's or Gavin's client
    my $sUsername      = $oCGI->param('username');
    my $sClientType     = $oCGI->param('client_type');
    my $sClientVersion = $oCGI->param('client_version');
    my $bFirstTime      = $oCGI->param('first_time');

    ##hash of NT info for current version of client
    ##version 1.0 is 0 in the array of upgrades
    my %infoNT;
    my @featuresNT;
    $infoNT{'current_version'} = '1.0';
    $infoNT{'force_upgrade'} = 0;
    $infoNT{'client_url'} = 'http://www.xdrive.com/download/xdrivent.exe';
    ##holds the first array subscript in which upgrade info is kept
    $infoNT{'1.0'} = 0;
    $featuresNT[0][0] = 'beta release';
    ## $featuresNT[0][1] = 'First new feature';
    ## $featuresNT[0][2] = 'Second new feature';

    ##hash of 95 info for current version of client
    ##version 2.03 is 0 in the array of upgrades
    my %info95;
    my @features95;
    $info95{'current_version'} = '2.03';
    $info95{'force_upgrade'} = 0;
    $info95{'client_url'} = 'http://www.xdrive.com/download/xdrive.exe';
    $info95{'2.00'} = 0;
    $info95{'2.01'} = 1;
    $info95{'2.02'} = 2;
    $info95{'2.03'} = 3;
    $info95{'2.04'} = 4;
    $features95[3][0] = 'automatic proxy support.';

    ## examples of other features
    ## $features95[0][1] = '2.03 feature 1';
    ## $features95[0][2] = '2.03 feature 2';
    ## $features95[1][0] = '2.04 feature 1';
    ## $features95[1][1] = '2.04 feature 2';

    my $returnString='';
    my $ref_to_hash;
    my $ref_to_array;

    ##point to hash and array for type of client
    ##this way no need to create separate functions
    if ($sClientType =~ /^xdwin9x/)
    {
        $ref_to_hash=\%info95;
    }
}
```

```

        $ref_to_array=\@features95;
    }
    elsif ($sClientType =~ /^xdwinnt/)
    {
        $ref_to_hash=\%infoNT;
        $ref_to_array=\@featuresNT;
    }
    else {}

    if (($sClientType =~ /^xdwin9x/) || ($sClientType =~ /^xdwinnt/))
    {
        ##if the user's version of the client is older than the
        ##current version, ask them to upgrade and tell them
        ##about new features
        my $feature_text='';
        if ($ref_to_hash->{'current_version'} > $sClientVersion)
        {
            ##get all features from the version 1 above the user's
            ##to the current version
            my $array_number_start = $ref_to_hash->{$sClientVersion} +
1;
            my $array_number_end = $ref_to_hash->{$ref_to_hash-
>{'current_version'}};
            ##Assemble a big string of new features for
            ##newer versions than user has
            my ($i,$j);
            for $i ($array_number_start .. $array_number_end)
            {
                for $j (0 .. @{$ref_to_array->{$i}})
                {
                    $feature_text .= " - ".$ref_to_array->{$i}{$j}
. "|";
                }
            }

            $returnString = join ("\n",
            "client_version=$ref_to_hash->{'current_version'}",
            "force_upgrade=$ref_to_hash->{'force_upgrade'}",
            "client_url=$ref_to_hash->{'client_url'}",
            "client_text=$feature_text",
            );
        }
        else
        {
            $returnString = join ("\n",
            "client_version=0.0",
            "force_upgrade=-1",
            "client_url=No url. Please contact X:drive",
            "client_text=",
            );
        }

        print $oCGI->header();
        print $returnString;

        ##if ($bFirstTime)
        ## {
        ##     ## Record the version number
        ##     ## XDClientFirstTimeUse
        ##     {
        ##         $sUsername,
        ##         $sClientType,
    
```

WO 01/33381

PCT/US00/30536

```
## $sClientVersion
## );
## }
```

###cookie.cgi

```
#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> to verify that the user is
# good to login, if they are then log them in and otherwise redirect to
# a not authorized page.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::UserQuota;
use XDrive::DatabaseO::Table::Language;
use XDrive::DatabaseO::Search;

use CGI;
use XDrive::CGI::Cookie;
use CGI::Carp qw(fatalsToBrowser);

use XDrive::CGI;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Template;
use XDrive::Library;
use XDrive::DatabaseO;
use Mail::Sendmail;

&main;
exit;

sub main
{
    my $oCGI      = new CGI;
    my $oErr      = new XDrive::Error;
    my $oDBO      = new XDrive::DatabaseO;
    my $oCookie   = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    my $oToken;
    my $sToken;
    my $sUsername;
    my $sPartnerCode;

    my $bSecurity = $oCGI->param('bSecurity');
    my $sPartnerToken = $oCGI->param('partner_token');

    my $passed_lang = $oCGI->param('language');

    #####
    ## Attempt to authenticate the user by using one of the following two
    ## authentication methods: username/password pair or partner token
    ## authentication.
    #####
    if (! defined $sUsername && length($sPartnerToken) > 20)
    {
        authPartnerUser($oCGI, $oErr, $oDBO, \$sUsername, \$oToken,
            \$sPartnerCode, $sPartnerToken);
        $sToken = $oToken->name();
    }
    else
    {
        authWebSiteUser($oCGI, $oErr, $oDBO, \$sUsername, \$oToken);
    }
}
```

```

        $sPartnerCode = 'xdrv';
    }

#####
## If an error occurred while trying to create a token then redirect
## the user to the error page.
#####
if ($oErr->Occurud)
{
    $oDBO->disconnect;
    xd_fatal_error($oCGI,$oErr);
    exit;
}

#####
## If we have gotten here then we have an authenticated user.
#####

#####
## Build and print out cookies
#####
my $sLanguage = getLanguage($oDBO,$sUsername);

##check if user's language is the same as passed language
if ((length($passed_lang) > 0) && $sLanguage ne $passed_lang)
{
    ##update db here to new language
    setLanguage($oDBO,$sUsername,$passed_lang);
    ##update session to new language
    $sLanguage = $passed_lang;
}

##delete the promo cookie; this will not be set here and we
##don't want an old one hanging out
##promo cookies should be set in promo.cgi
$oCookie->deleteElement('promo') if $oCookie->getElement('promo');

$oCookie->setElement
(
    (
        'language' => $sLanguage,
        'partner' => $sPartnerCode,
    ));

print "Set-Cookie: " . $oCookie->asString();
print "Set-Cookie: SST=$sToken; domain=.xdrive.com; path=/\n"
    if $sPartnerCode ne 'xdrv';

#####
## write user login to the database
#####
&incrementLoginNumber($oDBO,$sUsername,$sLanguage,$sPartnerCode);

#####
## Send the user off into thier file explorer
#####
if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
{
    print $oCGI->redirect("sst=".$sToken->name()."&sid=0");
}
else

```

```

        {
            xd_web_open($oCGI, "", "", \%ENV, $bSecurity);
        }

        $oDBO->disconnect;
        return 0;
    }

sub incrementLoginNumber()
{
    my $oDBO = shift;
    my $sUsername = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, $oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $timesLoggedIn = $oDiskAccount->fetchColumn("LOGIN_NUM");
    my $user_seq = $oDiskAccount->fetchColumn("USER_SEQ");

    if ($timesLoggedIn)
    {
        $timesLoggedIn++;
    }
    else
    {
        $timesLoggedIn=1;
    }

    $oDiskAccount->setColumn("LOGIN_NUM", $timesLoggedIn);
    $oDiskAccount->setColumn("LAST_LOGIN", XDToday());

    my $status = $oDiskAccount->update();

    if ($status > -1)
    {
        $oDiskAccount->commit();
        $oDiskAccount->finish();

        ##give user extra 10MB if 10th login
        if ($timesLoggedIn == 10)
        {
            my $oUserQuota = XDrive::DatabaseO::Table::UserQuota-
>new(undef, $oDBO);
            $oUserQuota->loadWhere("USER_SEQ", $user_seq);
            my $additional_quota = $oUserQuota-
>incrementQuota($user_seq, 10240);
            if ($additional_quota > 0)
            {
                &send_email($user_seq, $oDBO,
$additional_quota, $sLanguage, $sPartnerCode);
            }
        }
    }
    else
    {
        $oDiskAccount->rollback();
    }
}
}

```

```
sub send_email
```

```
{
    my $user_seq = shift;
    my $oDBO = shift;
    my $additional_quota = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    ##comes in as k, change to megabytes
    my $mbs = $additional_quota/1024;

    my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef,$oDBO);
    $oUserData->loadWhere("SEQ", $user_seq);
    my $email_address = $oUserData->fetchColumn("EMAIL_ADDRESS");
    my $name_first = $oUserData->fetchColumn("NAME_FIRST");
    my $name_last = $oUserData->fetchColumn("NAME_LAST");

    my $oTemplate = new XDrive::Template
    (
        (
            'language'      => $sLanguage,
            'partner_code' => $sPartnerCode,
        ));

    $oTemplate->load('received_10MB_10logins.thtml');
    $oTemplate->tags(
        (
            'mbs' => $mbs,
        ));
    $oTemplate->clear();
    my $message = $oTemplate->get;

    my %toXdrive =
    (
        To      => "$name_first $name_last <$email_address>",
        Bcc      => '',
        From     => "support\@xdrive.com",
        Message  => $message,
        Subject  => "Congratulations!"
    );

    sendmail(%toXdrive);
}
```

```
sub authPartnerUser
```

```
{
    my $oCGI = shift;
    my $oErr = shift;
    my $oDBO = shift;
    my $rsUsername = shift;
    my $roToken = shift;
    my $rsPartnerCode = shift;
    my $sPartnerToken = shift;

    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    my $oPartnerToken = new Token
    (
        (
            'err' => $oErr,
            'dbh' => $oDBO,
        ));
    $oPartnerToken->load($sPartnerToken);

    return if $oErr->Occurud;
```



```

    $$roToken = new Token
        (
            'dbh' => $oDBO,
            'err' => $oErr,
            'user_sequence' => $oPartnerToken->data('user_seq'),
        );
    $$roToken->create();

    return if $oErr->Occurud;

    ### Edited by Justin so that the partner_code is looked for
    ### in the cookie instead of the token table.
    $$rsPartnerCode = $oPartnerToken->data('partner_code');
    ##$$rsPartnerCode = $oCookie->getElement('partner');
    $$rsUsername = $oPartnerToken->data('user');

    $$roToken->data('ip', $ENV{REMOTE_ADDR});
    $$roToken->data('browser', $ENV{HTTP_USER_AGENT});
    $$roToken->data('user', $$rsUsername);
    $$roToken->data('user_seq', $oPartnerToken->data('user_seq'));
    $$roToken->data('partner_code', $$rsPartnerCode);
    $$roToken->data('disk_account_seq', $oPartnerToken-
>data('disk_account_seq'));
    $$roToken->save;

    $oPartnerToken->delete();
}

sub authWebSiteUser
{
    my $oCGI = shift;
    my $oErr = shift;
    my $oDBO = shift;
    my $rsUsername = shift;
    my $roToken = shift;

    my $sPassword = $oCGI->param('pass');
    $$rsUsername = $oCGI->param('user');

    $oCGI->param('user');

    if (xd_auth_password($$rsUsername, $sPassword, $oDBO))
    {
        ## Login the user info X:drive and get the session token
        $$roToken = xd_login($oCGI, $$rsUsername, $oErr, $oDBO);
    }
    else
    {
        $oErr->AddErrorByErrorCode('501');
    }
}

sub getLanguage
{
    my $oDBO = shift;
    my $sUsername = shift;

    my $language;

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, $oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);

```

```

    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::Database0::Table::UserSettings-
>new(undef,$oDBO);
    $oUserSettings->loadWhere("USER_SEQ",$userSeq);
    $oUserSettings->finish;
    my $language = $oUserSettings->fetchColumn("LANGUAGE");

    if ($language eq '')
    {
        $language = 'english';
    }
    else
    {
        ## Get language from database given code
        my $oLanguage = XDrive::Database0::Table::Language-
>new(undef,$oDBO);
        $oLanguage->loadWhere("SEQ",$language);
        $oLanguage->finish;
        $language = $oLanguage->fetchColumn("CODE");
    }

    return $language;
}

sub setLanguage
{
    ##set the LANGUAGE column of the User_Settings table to passed
    language

    my $oDBO = shift;
    my $sUsername = shift;
    my $language = shift;

    my ($rv,$errorCode);

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::Database0::Table::DiskAccount-
>new(undef,$oDBO);
    ##grab right table
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::Database0::Table::UserSettings-
>new(undef,$oDBO);
    $oUserSettings->loadWhere("USER_SEQ",$userSeq);
    $oUserSettings->finish;

    ##grab the seq number of the LANGUAGE being passed
    my $oLanguage = XDrive::Database0::Table::Language->new(undef,$oDBO);
    $oLanguage->loadWhere("CODE", $language);
    $oLanguage->finish();
    my $seq_lang = $oLanguage->fetchColumn("SEQ");

    eval
    {
        ##
        ##set language here
        $rv = 0;
    }
}

```

```
        $oUserSettings->setColumn('LANGUAGE',$seq_lang);
        $rv = $oUserSettings->update();
    };
    if ($rv == 0)
    {
        $oUserSettings->rollback();
        $errorCode = 0;
    }
    else
    {
        $oUserSettings->commit();
        $errorCode = 1;
    }
    return $errorCode;
}
```

###download_client.cgi

```

#!/usr/bin/perl
## Written by Karen Eppinger
## Script that shows the 'download the client' page
## it can no longer be static html because we need to
## do some checking on whether the user is from a partner or not
## if so, make sure to let them know what their X:drive login name
## is if it differs from their partner login

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use XDrive::Library;
use XDrive::Template;
use XDrive::Error;
use XDrive::DatabaseO;
use XDrive::Client::Security;
use XDrive::DatabaseO::Table::ResellerUserMap;
use XDrive::DatabaseO::Table::Reseller;

&main;
exit;

sub main
{
    ## Load the session token
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;
    my $oCGI = new CGI;

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        $oDBO->disconnect();
        exit;
    }

    my $partner_code = $oToken->data('partner_code');
    my $language = $oCookie->getElement('language') || 'english';

    my $oForm = new XDrive::Template
    (
        'partner_code' => $partner_code,
        'language' => $language,
    );

    $oForm->load('download_client.thtml');

    ##if we are coming from a partner, make sure partner login
    ##and X:drive login match

    my $reseller_username;
    my $reseller_name;
    my $partner_warning;
    my $username;

```

```

    if ($partner_code ne 'xdrv')
    {
        my $user_seq = $oToken->data('user_seq');
        $username = $oToken->data('user');

        my $oResellerUserMap = XDrive::DatabaseO::Table::ResellerUserMap-
>new(undef, $oDBO);
        my $oReseller = XDrive::DatabaseO::Table::Reseller->new(undef,
$oDBO);

        $oReseller->loadWhere("CODE", $partner_code);
        $reseller_name = $oReseller->fetchColumn("NAME");
        $oResellerUserMap->loadWhere("USER_SEQ", $user_seq);
        $reseller_username = $oResellerUserMap->fetchColumn("ALIAS");

        if ($reseller_username ne $username)
        {
            ##load the text for the warning message
            my $oWarning = new XDrive::Template
                (
                    'partner_code' => $partner_code,
                    'language' => $language,
                );

            $oWarning->load('download_client_warning.shtml');
            $oWarning->tags
                (
                    'reseller_name' => $reseller_name,
                    'reseller_username' => $reseller_username,
                    'username' => $username,
                );

            $oWarning->clear();
            $partner_warning = $oWarning->get();
        }
    }

    $oForm->tags
    (
        'partner_warning' => $partner_warning,
        'reseller_name' => $reseller_name,
        'reseller_username' => $reseller_username,
        'username' => $username,
    );

    $oForm->clear();

    print $oCGI->header(), $oForm->get;

    $oDBO->disconnect();

    return 0;
}

```

###email_change.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::DiskAccount;

use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Library;
use XDrive::Template;
use XDrive::Security;
use XDrive::CGI;
use XDrive::Error;

use strict;

&main;
exit;

sub main
{
    my $oCGI = CGI->new();
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    #####
    ## Check the token is valid and is an error occurred then
    ## redirect with a fatal error
    #####

    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    my $sUserName = $oToken->data('user');
    my $sOldEmail = $oCGI->param('oldEmail');
    my $sNewEmail = $oCGI->param('newEmail');

    if (($sOldEmail eq '') || ($sNewEmail eq ''))
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1350);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    ##first, get user_seq from the disk_account table
    ##since we only have the user name, need to do this first
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,undef);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);
    my $nUserID = $oDiskAccount->fetchColumn('USER_SEQ');

    ##now that we have that, get the email address from
```

```

##user table using the user_seq number to pull the seq number
my $oUserInfo = XDrive::Database0::Table::UserData->new(undef,
$oDiskAccount->fetchDBO());
$oUserInfo->loadWhere('EMAIL_ADDRESS', $sNewEmail);

##if a sequence number is returned, there is already a record
##in the database with that email address. don't allow to change

my $nSeqNumber = $oUserInfo->fetchColumn('SEQ');

if ($nSeqNumber)
{
    $oUserInfo->disconnect();
    my $sMessage = $oErr->ReturnMessageGivenCode(1351);
    XDErrorToBrowser("", $sMessage, undef, $oToken);
}
else
{
    $oUserInfo->loadWhere('SEQ', $nUserID);
    my $sEmailinDB = $oUserInfo->fetchColumn('EMAIL_ADDRESS');

    if ($sOldEmail eq $sEmailinDB)
    {
        ##set email in class
        $oUserInfo->setColumn('EMAIL_ADDRESS', $sNewEmail);
        ##now update database
        $oUserInfo->update();

        my $oTemplate = new XDrive::Template
            ({'partner_code' => $oToken-
>data('partner_code')});
        $oTemplate->load('pr_changeemail_ok.shtml');
        print "Content-type: text/html\n\n";
        print $oTemplate->get();
    }
    else
    {
        $oUserInfo->disconnect();
        my $sMessage = $oErr->ReturnMessageGivenCode(1352);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }
}
$oUserInfo->commit();
$oUserInfo->finish();
$oUserInfo->disconnect();
}

```

###error.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});
use XDrive::Error;
use XDrive::Template;
use CGI;

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my ($sErrorCode) = $ENV{QUERY_STRING} =~ /error=([^&\=]+)/;
    my $oError = new XDrive::Error;
    my $sError = $oError->ReturnMessageGivenCode($sErrorCode);
    my $oTemplate = new XDrive::Template( {'partner_code' => 'xdrv'} );
    $oTemplate->load('generic_error.shtml');
    $oTemplate->tags
        (
            {
                'message' => $sError
            }
        );
    $oTemplate->clear();
    print $oCGI->header(), $oTemplate->get;
}
```


###explorer.cgi

```

#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Tue May 25 15:23:31 PDT 1999.
## Program to build the file explorer which is itself a popup window.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
#use vars qw(@ISA);

#@ISA = qw(XDrive::CGI);

use CGI qw(param header);
use CGI::Carp qw(fatalsToBrowser);
use Date::Format;
use HTTP::Icons;
# use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::Client::Quota;
use XDrive::Library;
use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::Error;

&main;

exit(0);

sub main
{
    #####
    ## Global variables
    #####
    my $oToken;          ## XDrive Token
    my $sUsername;       ## username
    my $sPath;           ## path for index
    my $sSST;            ## Token name
    my $bEditExt;        ## Allow extensions to be edited?
    my $bFirstTime;      ## First time the've logged in...
    my $bExtraHelp;      ## Print extra help
    my $bMarketing;      ## does user want to receive offers from other
companies
    my $bNewsletter;     ## does user want to receive our newsletter
    my $sPartner;        ## partners name
    my $g_sFrameSize;    ## breakdown of the centerview frame
    my $g_sFrameBanner;  ## banner view frame information

    my $oDBO = XDrive::DatabaseO->new(undef,undef);
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    #####
    ## If the user has bookmarked the X:drive service then redirect
    ## them back to the homepage
    #####

```

```

    if (! length($oCGI->param('sst')) && ! length($oCGI->cookie('SST')))
    {
        print $oCGI->redirect('/cgi-bin/web_unauthorized.cgi?error=804');
        $oDBO->disconnect();
        return 0;
    }

    ####
    ## Check the security and if an error occurs
    ####
    $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    if ($oErr->Occurud) {
        $oDBO->disconnect();
        xd_fatal_error($oCGI, $oErr);
        exit;
    }

    ####
    ## Now we know we have a valid session so pull the partner name
    ## from a cookie if available or clear the variable
    ####
    # $sPartner = $oToken->data('partner_code');
    $sPartner = $oCookie->getElement('partner');
    $sPartner = "xdrv" if ($sPartner eq "");

    ## Load the required template HTML files.
    # my $oFrame = new XDrive::Template
    #     ({
    #         'partner_code' => $oToken->data('partner_code')
    #     });

    #### Edited by Justin to check the cookie instead of
    #### the token table for the partner_code.
    my $oFrame = new XDrive::Template
        ({
            'partner_code' => $oCookie->getElement('partner')
        });

    ## If the request comes from the windows app the give back a simplified
    template
    $oFrame->load("acct_explorer_frame.shtml");

    ## Assign globally used variables
    $sPath = $oCGI->param('sFolderCurrent');
    $sSST = $oToken->name;
    $sUsername = $oToken->data('user');

    ## User settings
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new(undef,
    $oDBO);

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings->new(undef,
    $oDBO);
    my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef, $oDBO);

    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oUserSettings->loadWhere("USER_SEQ", $oDiskAccount-
    >fetchColumn("USER_SEQ"));

```

```

$UserData->loadWhere("SEQ", $oDiskAccount->fetchColumn("USER_SEQ"));

$bEditExt      = $oUserSettings->fetchColumn("FILE_EXT_EDITABLE") == 1 ?
'true' : 'false';
$bExtraHelp    = ($oUserSettings->fetchColumn("EXTRA_HELP") == 1) ?
'true' : 'false';
$bMarketing    = $oUserSettings->fetchColumn("OPT_MARKETING") == 1 ?
'true' : 'false';
$bNewsletter   = $oUserSettings->fetchColumn("OPT_NEWSLETTER") == 1 ?
'true' : 'false';

my $firstName = $UserData->fetchColumn("NAME_FIRST");
my $lastName  = $UserData->fetchColumn("NAME_LAST");

my $first = $oCGI->param('first');
$bFirstTime = $first eq 'yes' ? 'true' : 'false';

## Frame settings
if ($sPartner eq 'cc' || $sPartner eq 'qupa')
{
    $g_sFrameSize = '100%';
    $g_sFrameBanner = '';
}
else
{
    $g_sFrameSize = '103,*';
    $g_sFrameBanner = '<FRAME NAME="banner"'.
        ' SRC="/cgi-bin/ads.cgi" SCROLLING=NO BORDER=0 ' .
        ' FRAMEBORDER=0 MARGINWIDTH=0 MARGINHEIGHT=0 ' .
        ' TOPMARGIN=0 LEFTMARGIN=0>';
}

##get the language information from the cookie
##if no cookie or not set, set to english
my %session_info = $oCGI->cookie('x_session_info');
my $language;

if ($session_info{'language'} ne '') {
    $language = $session_info{'language'};
}
else {
    $language = 'english';
}

my $clientDownload = $oCGI->param('client');
my $sCenterPage = 'centerview.shtml';
if ($clientDownload eq 'getclient') {
    $sCenterPage = 'download_client.shtml';
}

## Set the token name and session ID in the navigation form so that
popup
## windows have access to them and the do not need to be passed around.
$oFrame->tags
(
    'sSST' => $sSST,
    'bSettingEditExtensions' => $bEditExt,
    'sPartner' => $sPartner,
    'bExtraHelp' => $bExtraHelp,
    'bFirstTime' => $bFirstTime,
    'bMarketing' => $bMarketing,
    'bNewsletter' => $bNewsletter,
    'centerPage' => $sCenterPage,

```

```
'userName' => $sUsername,  
'firstName' => $firstName,  
'lastName' => $lastName,  
'frameBanner' => $g_sFrameBanner,  
'frameSize' => $g_sFrameSize,  
'language' => $language  
));  
  
## Print out the HTML and exit  
$oFrame->clear();  
print $oCGI->header(), $oFrame->get;  
  
$oDiskAccount->finish();  
$oUserSettings->finish();  
$oUserData->finish();  
  
$oDBO->disconnect();  
}
```

###explorer_user_data.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Tue May 25 15:23:31 PDT 1999.
## Program to build the file explorer which is itself a popup window.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use vars qw(@ISA);

@ISA = qw(XDrive::CGI);

use Data::Dumper;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use Token;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::Library;
use XDrive::Template;
use XDrive::Error;

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    my $sFolder;
    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);

    $sFolder = $oCGI->param('folder_current');

    ## Load the required template HTML files.
    my $oFrame = new XDrive::Template
    (
        {
            'partner_code' => $oToken->data('partner_code')
        }
    );

    if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
    {
        $oFrame->load("acct_user_data_xd_win.thtml");
    }
    else
    {
        $oFrame->load("acct_user_data.thtml");
    }
}
```

```
## Set the token name and session ID in the navigation form so that
popup
## windows have access to them and the do not need to be passed around.
$oframe->tags
(
  'sst' => $oaction->SST(),
  'sid' => $oaction->SID(),
  'usage_total' => $oaction->QuotaLimit(),
  'usage_used' => $oaction->QuotaUsed(),
  'stuff' => $oaction->DiskAccountXML($sFolder)
);
$oframe->clear;

$oaction->DisconnectDB();

## Print out the HTML and exit
print "Cache-Control: no-cache\n";
print "pragma: no-cache\n";
print "Content-type: text/html\n\n";
print $oframe->get;
}
```

###file_load.cgi

```

#!/usr/bin/perl
# Program written by Martin Hald <mhald@uci.edu> to fetch files from a
# storage area or database and return them via a HTTP socket to the user.

use strict;
use CGI qw(header param);
use CGI::Carp 'fatalsToBrowser';

## The HTTP::MimeTypes module was a quick module that I wrote that reads the
## standard apache mime.types file, parses it and given any known extension
## translates it to the correct mimetype.

use lib ($ENV{PERL_XDRIVE_LIB});

use HTTP::MimeTypes;
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::DatabaseO::Table::DiskItemShare;
use XDrive::DatabaseO;
use XDrive::Library;
use XDrive::Error;

## We have two security methods when downloading files:
## 1) tokens
## 2) claim checks
## to deal with this we simply security method we are using and process the
## request.

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    my $sFileCurrent; ## Current File
    my $oAction;      ## Action object

    ####
    ## Process the request as a share a file pickup if the claim_check
    ## param is available
    ##
    if (param('claim_check'))
    {
        my $oShare;
        $oShare = XDrive::DatabaseO::Table::DiskItemShare->new();
        $oShare->loadWhere("random_key", param('claim_check'));

        $oAction = new XDrive::Client::Actions($oShare,$oCGI);
        $sFileCurrent = join
        (
            '/',
            $oShare->fetchColumn("ITEM_PATH"),
            $oShare->fetchColumn("ITEM_NAME")
        );
    }
    ####
}

```

```

## Otherwise it is an request from the browser or PC client
## side program so let the actions object handle the request
####
else
{
####
## Attempt to authenticate the user and if that fails
## then redirect to the error CGI
####
my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

if ($oErr->Occurud)
{
xd_fatal_error($oCGI,$oErr);
exit;
}

####
## Now we know that we have a valid token so go ahead
## and let the actions object handle the request
####

$oAction = new XDrive::Client::Actions
(
$oToken,
$oCGI
);

$sFileCurrent = $oAction->FileCurrent();
}

## Check that the current file is OK. If this check fails then
## the code does an XLErrorToBrowser and exists
$oAction->FileCheck($sFileCurrent);

print _header($sFileCurrent);

## Commented out by Justin because it was
## including a 1 at the end of the file by printing it out.
#print $oAction->FileLoad($sFileCurrent);
$oAction->FileLoad($sFileCurrent);

$oDBO->disconnect;
}

sub _header
{
my $sFile = shift;
my $mlt = new HTTP::MimeTypes;

## Grab the extension and lookup the correct mimetype using the mlt or
mime ## lookup table object.

my $sHeader; ## MIME header
my $sExtension; ## file extension

## Clean up the filename by getting rid of any path that comes before
## the filename.
$sFile =~ s=.*/==g;

if (param('mime') eq 'download')

```



```

        {
            if ($ENV{HTTP_USER_AGENT} =~ /MSIE/)
            {
                $sHeader .= "Content-Disposition: attachment;
filename=$sFile\n";
                $sHeader .= "Content-type: application/download;
name=\"\$sFile\"\\n\\n";
            }
            else
            {
                $sHeader .= "Content-type: application/octet-stream\\n\\n";
            }
        }
    else
    {
        my $dotPos=-1;
        my $returnPos=-1;
        while (($dotPos = index($sFile, ".", $dotPos)) > -1)
        {
            $returnPos = $dotPos;
            $dotPos++;
        }

        ##if no extension set extension to nothing
        if ($returnPos < 0)
        {
            $sExtension='';
        }
        else
        {
            $sExtension = substr($sFile,$returnPos+1);
        }

        $mlt->extension($sExtension);
        $sHeader = $mlt->header();
    }

    return $sHeader;
}

sub IEHack ()
{
    my $sFileCurrent = param('sFileCurrent');
    my ($sFileOnly) = $sFileCurrent =~ /\\[^\[\]]+\$/;
    my $sJavascript;

    if (param('source') eq 'www.fileExplorer.view' || param('source') eq
'www.fileExplorer.download')
    {
        $sJavascript = <<EOM;
<SCRIPT LANGUAGE=JAVASCRIPT>
if (parent.parent.parent.name)
{
    parent.parent.parent.parent.XDReset();
    parent.parent.parent.parent.XDRefreshExplorer();
}
</SCRIPT>
EOM
    }

    print <<EOM;
Content-type: text/html

```

<HTML>

<BODY>

\$sJavascript

```
<OBJECT classid=CLSID:4CCF6192-4552-11D3-80A8-0050048D4BF8
        codebase="http://209.101.43.96/dll/xdfiles.cab"
        id=XDFiles>
</OBJECT>
```

<SCRIPT LANGUAGE="VBSCRIPT">

```
' Don't raise errors
On Error Resume Next
```

```
Dim oXDFiles          ' The ActiveX control
```

```
' Late bind to the control
Set oXDFiles = CreateObject("XDFiles.XDFiles.1")
```

```
' If we got an error, they didn't install the ActiveX control
If Err.Number <> 0 Then
    MsgBox "You must install the X:drive ActiveX control in order to
download " &
        "the file. Please click Download again and when prompted to
install the " &
        "ActiveX control, click Yes."
End If
```

```
' Set some test values for the properties
oXDFiles.Prompt = True
oXDFiles.Destination = "c:\\$sFileOnly"
oXDFiles.File = "$sFileCurrent"
```

```
' Call each method
oXDFiles.Get
```

```
' Print out each property
' document.write("oXDFiles.Destination = " & oXDFiles.Destination & "<br>")
' document.write("oXDFiles.Prompt = " & CBool(oXDFiles.Prompt) & "<br>")
' document.write("oXDFiles.File = " & oXDFiles.File & "<br>")
' document.write("oXDFiles.ServerSideToken = " & oXDFiles.ServerSideToken &
"<br>")
' document.write("oXDFiles.SessionId = " & oXDFiles.SessionId & "<br>")
```

```
' Free the ActiveX control
Set oXDFiles = Nothing
```

</SCRIPT>

</BODY>

</HTML>

EOM

}

###file_save.cgi

```

#!/usr/bin/perl

#####
### file_save.cgi
#####

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
use CGI::Carp qw(fatalsToBrowser);
use Token;

use XDrive::CGI2;          ## file upload functions
use XDrive::CGI qw(:MAIN); ## xd_web_buttonindex function
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Library;       ## xd_fatal_error function
use XDrive::DatabaseO;
use XDrive::DatabaseO::Search;
use XDrive::Template;
use XDrive::DatabaseO::Transaction;

&main;

exit;

sub main {
    my $oErr    = new XDrive::Error;
    my $oDBO    = new XDrive::DatabaseO;
    my $oSearch = new XDrive::DatabaseO::Search;

    my $oTransaction = XDrive::DatabaseO::Transaction->new($oDBO);

    #####
    ## Parse the SST cookie manually and retrieve the user sequence
    ## by passing it to the getUserSeq sub.
    #####
    my ($cookie) = $ENV{'HTTP_COOKIE'} =~ /\bSST=(\w+)\b/;
    my $user_seq = &getUserSeq($oSearch, $cookie);

    my $bytes = $ENV{'CONTENT_LENGTH'}; ## number of bytes being uploaded.

    my %upload_hash = ('USER_SEQ' => $user_seq,
                       'BYTES'    => $bytes);
    my $oCGI = new XDrive::CGI2(\%upload_hash, $oTransaction);

    #####
    ## Attempt to authenticate the user and if the authentication
    ## fails then redirect to the error CGI
    #####
    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    if ($oErr->Occurud){
        xd_fatal_error($oCGI, $oErr);

        exit;
    }
}

```

```
#####
####
#####
### Check to see if they've exceeded
### their quota limit, and error if so.
#####
# my $oUserQuota = XDrive::DatabaseO::Table::UserQuota->new(undef,
$oDBO);
# $oUserQuota->loadWhere("USER_SEQ", $user_seq);
# my $nQuota = $oUserQuota->fetchColumn("QUOTA");
# my $nDiskUsed = $oUserQuota->fetchColumn("DISK_USED");
# if ( ($nQuota * 1024) < ($nDiskUsed + $bytes) ) {
#     $oUserQuota->finish();
#     $oDBO->disconnect();
#     ## let user know he or she has exceeded his quota
#     $oErr->AddErrorByErrorCode(1240);
#     XDErrorToBrowser('action_upload_error.shtml', $oErr, 1,
$oToken);
#     exit(0);
# }
#####
####
####
## Authentication succeeded so we have a valid session, let
## the actions object handle the request
####
my $oAction = new XDrive::Client::Actions($oToken, $oCGI);

$oAction->SaveUploadedFiles();

####
## File has been uploaded at this point, so set
## the upload inactive in the database.
####
$oTransaction->setUploadInactive();

xd_web_buttonindex($oCGI);
$oAction->DisconnectDB();

$oSearch->disconnect();

return 0;
)

#####
### Subroutine:  getUserSeq
### Parameters:  one object, one scalar
### Returns:    one scalar
### Description: Receives a database search object and an SST token.
###             Queries the token table for the user sequence and returns
###             it.
#####
sub getUserSeq ($$) {
    my $oSearch = shift;
    my $sst_code = shift;

    my $st = "SELECT user_seq FROM token WHERE code = '$sst_code'";

    my $data = $oSearch->XDSQLSearch($st);

    return $$data[0][0];
}

```

#! file_upload_stat.cgi

```

#!/usr/bin/perl

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use XDrive::CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::DatabaseO;
use XDrive::DatabaseO::Search;
use XDrive::Error;
use XDrive::Client::Security;
use XDrive::Template;
use XDrive::Library;
use Token;

&main();

exit(0);

sub main {
    my $oCGI = new CGI;
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    #####
    ### Security Check
    #####
    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);
    if ($oErr->Occurud || (! $oToken)) {
        XDErrorToBrowser("", "Security Violation: No token", undef, $oToken);
    }

    my ($tmp_file, @stat_array, $stat_bytes, $meta_refresh, $percent,
        $width_green, $width_red);
    my ($url, $tmp_file_string, @tmp_file_array, $error_code);

    my $tmp_path = XDFileUploadTempDir;

    my $oTemplate = new XDrive::Template( {'partner_code' => 'xdrv',
                                           'file' =>
'file_upload__status.shtml'} );

    my $id                = $oCGI->param('id');          ## unique
upload id
    my $nof                = $oCGI->param('nof');          ## number of
files
    my $nof_queried        = $oCGI->param('nof_queried');  ## nof
retrieved from db
    my $file_param         = $oCGI->param('tmp_file');     ## initial file
string
    my $total_file_string = $oCGI->param('total_file_string'); ## string of
all files
    my $param_uploaded     = $oCGI->param('uploaded');     ## bytes
uploaded
    my $bytes              = $oCGI->param('bytes');        ## total number
of bytes

    print $oCGI->header();

```

```

#####
### First, if we're passed an upload id and no temp file params (files to
stat),
### then we either haven't queried the database yet and need to or need to
### query the database again because the number of files (nof) being
uploaded
### is greater than the number of files that our first database query
returned.
#####

if ($id && (! $file_param)) {
. ### If this is the first pass, then percent will be a space and width
will be 0.
    $percent = $bytes ? int(100 * ($param_uploaded / $bytes)) : '&nbsp;';
    $width_green = ($percent eq '&nbsp;') ? 0 : $percent;
    $percent .= '%' unless $percent eq '&nbsp;';

    my $seconds;

    $width_red = &width_red($width_green);

    $oTemplate->tags( {'width_green' => $width_green,
                      'width_red'   => $width_red,
                      'percent'     => $percent});

    my $oSearch = new XDrive::DatabaseO::Search($oDBO);

    my ($cnt, $data) = $oSearch->uploadStatusSearch($id);

    ### If no rows were returned from the database, then redirect
    ### and re-query the database.
    if ($cnt == 0) {
        $oSearch->disconnect();

        $seconds = 0;

        $url = "/cgi-bin/file_upload_stat.cgi?" .
            "id=$id&nof=$nof&bytes=$bytes&uploaded=$param_uploaded";

        $meta_refresh = &buildMetaRefresh($seconds, $url);

        &connectingToServer($meta_refresh, $oTemplate);

        exit(0);
    }
    else {
        my $i = 0;

        $bytes      = $$data[$i][0];
        $error_code = $$data[$i][2];

        foreach (@$data) {
            $tmp_file = $$data[$i][1];

            push @tmp_file_array, $tmp_file;

            $i++;
        }

        $tmp_file_string = join '~', @tmp_file_array;

        if ($cnt == $nof) {
            $oSearch->disconnect();

```

```

        &statFilesTotal($bytes, $tmp_file_string, $oTemplate);
    }
    exit(0);
}

$seconds = 0;

$url = "/cgi-bin/file_upload_stat.cgi?" .
    "id=$id&nof=$nof&uploaded=$param_uploaded&" .
    "nof_queried=$cnt&bytes=$bytes&tmp_file=$tmp_file_string";

$meta_refresh = &buildMetaRefresh($seconds, $url);

my $bytes_uploaded = ($param_uploaded > 0) ? $param_uploaded : '-';

&redirect($meta_refresh, $bytes_uploaded, $bytes, $oTemplate);

$oSearch->disconnect();

exit(0);
}
}
elseif ($file_param) {
    $oDBO->disconnect();

    my @file_array = split '~', $file_param;
    my $ary_cnt = @file_array;

    my ($uploaded_bytes, $seconds);

    if (scalar @file_array > 0) {
        foreach (@file_array) {
            @stat_array = stat("$tmp_path/$_");
            $stat_bytes = $stat_array[7];

            $uploaded_bytes += $stat_bytes;
            push @tmp_file_array, $_;
        }
        if ( ($uploaded_bytes == $param_uploaded) && ($nof > $nof_queried) )
        {
            $seconds = 0;

            $url = "/cgi-bin/file_upload_stat.cgi?" .
                "id=$id&nof=$nof&bytes=$bytes&uploaded=$param_uploaded";

            $meta_refresh = &buildMetaRefresh($seconds, $url);

            $percent = ($bytes == 0) ? 0 : int(100 * ($param_uploaded /
$bytes));
            $width_green = $percent;
            $percent .= '%';

            &redirect($meta_refresh, $uploaded_bytes,
                $bytes, $oTemplate, $percent, $width_green);

            exit(0);
        }
        else {
            $tmp_file_string = join '~', @tmp_file_array;
        }
    }
}

```

```

$percent = ($bytes == 0) ? 0 : int(100 * ($uploaded_bytes / $bytes));
$width_green = $percent;
$percent .= '%';

$percent = '&nbsp;' if $width_green == 0;

$seconds = 2;

$url = "/cgi-bin/file_upload_stat.cgi?" .
      "id=$id&bytes=$bytes&nof=$nof&nof_queried=$nof_queried&" .
      "uploaded=$uploaded_bytes&tmp_file=$tmp_file_string";

$meta_refresh = &buildMetaRefresh($seconds, $url);

&redirect($meta_refresh, $uploaded_bytes, $bytes, $oTemplate, $percent,
$width_green);

exit(0);
}
elseif ($total_file_string) {
    $oDBO->disconnect();
    &statFilesTotal($bytes, $total_file_string, $oTemplate);
}
else {
    $oDBO->disconnect();

    &closeWindow($oTemplate);

    exit(0);
}
}

sub statFilesTotal ($$$) {
    my ($bytes, $tmp_file_string, $oTemplate) = @_;

    my $tmp_path = XDFileUploadTempDir;

    my @file_array = split '~', $tmp_file_string;

    my (@tmp_file_array, $uploaded_bytes, @stat_array, $stat_bytes);

    my $file_cnt = 0;

    foreach (@file_array) {
        if (-e "$tmp_path/$_") {
            @stat_array = stat("$tmp_path/$_");
            $stat_bytes = $stat_array[7];

            $uploaded_bytes += $stat_bytes;

            push @tmp_file_array, $_;

            $file_cnt++;
        }
    }

    if ($file_cnt == 0) {
        &closeWindow($oTemplate);

        exit(0);
    }
    else {
        my $percent = int(100 * ($uploaded_bytes / $bytes));

```



```

    my $width_green = $percent;

    $percent .= '%';

    $percent = '&nbsp;' if $width_green == 0;

    my $seconds = 2;

    my $url = "/cgi-bin/file_upload_stat.cgi?" .
        "bytes=$bytes&total_file_string=$tmp_file_string";

    my $meta_refresh = &buildMetaRefresh($seconds, $url);

    &redirect($meta_refresh, $uploaded_bytes, $bytes, $oTemplate, $percent,
$width_green);

    exit(0);
}

sub redirect ($$$$;$) {
    my ($meta_refresh, $bytes_uploaded, $bytes, $oTemplate, $percent,
$width_green) = @_;

    if ($bytes > 1024) {
        $bytes = sprintf "%.f", ($bytes / 1024);
        $bytes .= 'k';
    }

    if ($bytes_uploaded > 1024) {
        $bytes_uploaded = sprintf "%.f", ($bytes_uploaded / 1024);
        $bytes_uploaded .= 'k';
    }

    my $width_red = &width_red($width_green);

    $oTemplate->tags( {'meta_refresh' => $meta_refresh,
                    'bytes_uploaded' => $bytes_uploaded,
                    'bytes_total' => $bytes,
                    'percent' => $percent,
                    'width_green' => $width_green,
                    'width_red' => $width_red} );

    $oTemplate->clear();

    print $oTemplate->get;
}

sub closeWindow ($) {
    my $oTemplate = $_[0];

    $oTemplate->load('file_upload_stat__window_close.shtml');

    print $oTemplate->get;
}

sub connectingToServer ($$) {
    my ($meta_refresh, $oTemplate) = @_;

    $oTemplate->load('file_upload__connecting.shtml');

    $oTemplate->tags( {'meta_refresh' => $meta_refresh} );
}

```

```
print $oTemplate->get;
}

sub buildMetaRefresh ($$) {
    my ($seconds, $url) = @_;

    my $meta_refresh = "<meta http-equiv=refresh content=\""$seconds;
url=$url\">";

    return $meta_refresh;
}

sub width_red {
    my $width_green = shift;
    my $width_red = ((100 - $width_green) > 0)? 100 - $width_green : 0;

    return $width_red;
}
```

###folder_create.cgi

```

#!/usr/bin/perl
# Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999.

use strict;
use vars qw(@ISA);
use lib ($ENV{PERL_XDRIVE_LIB});
#use lib qw(/export/home/xdrive/lib);

$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)); # Make %ENV safer

@ISA = qw(XDrive::CGI);

use CGI::Carp 'fatalsToBrowser';
use Date::Format;
use Token;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::Client::Actions;

use CGI;
use XDrive::DatabaseO;
use XDrive::Error;

&main;
exit;

sub main
{
    my $oCGI = new CGI;
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    #####
    ## Attempt to authenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If the authentication failed then redirect to the
    ## error cgi and exit
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    #####
    ## Otherwise we know that we have a valid session and
    ## can continue normally
    #####

    my $oAction = new XDrive::Client::Actions
    (
        $oToken,
        $oCGI
    )

```

```
    );  
  
    $oAction->FolderCreate();  
    xd_web_buttonindex($oCGI);  
    $oAction->DisconnectDB();  
    return 0;  
}
```

###forgot_password.cgi

```

#!/usr/bin/perl

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(param header);
use CGI::Carp qw(fatalsToBrowser);
use Token;
use XDrive::CGI ();
use XDrive::Template;
use XDrive::Client::Registration;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Search;
use XDrive::Library;
use XDrive::Utils::RandomString;

use Mail::Sendmail;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;

#####
my $request_template = "forgot_password_request.shtml";
my $thank_you_template = "forgot_password_t_y.shtml";
my $alert_template = "forgot_password_alert.shtml";
my $email_template = "password_admin_email.shtml";
#####

exit &main();

sub main {
    my $oCGI = CGI->new();

    my $sEmailAddress = $oCGI->param('txtEmailAddress');
    my $sUsername = $oCGI->param('txtUsername');

    my $oContent = new XDrive::Template( {'partner_code' => 'xdrv' } );
    my $oNavigation = new XDrive::Template( {'partner_code' => 'xdrv' } );
    my $oLayout = new XDrive::Template( {'partner_code' => 'xdrv' } );

    ## Load the required template HTML files.
    $oNavigation->load("front_nav.shtml");
    $oContent->load("front_signup.shtml");
    $oLayout->load("layout.shtml");

    if ( ($sEmailAddress) && ($sUsername) ) {
        ## Change user's password
        my @characters = ('a'..'z', 'A'..'Z', '0'..'9');
        my $sRandomKey = XDRandomString(8, \@characters);
        if(&PasswordSet($oContent, $sUsername, $sEmailAddress, $sRandomKey)) {
            sendMail($oContent, $sUsername, $sRandomKey, $email_template);
        }
        &display_form($oContent, $thank_you_template);
    } else {
        &display_form($oContent, $request_template);
    }
}

```

```

    }

    ## Print out the HTML and exit
    $oLayout->tags
        (
            'header_graphic' => 'header_fill.gif',
            'title' => 'What is my password?',
            'content' => $oContent->get,
            'navigation' => $oNavigation->get,
        );
    $oLayout->clear;

    print header, $oLayout->get;
    return 0;
}

sub PasswordSet
{
    my($oContent, $sUsername, $sEmailAddress, $sPassword) = @_;
    my $bReturnValue = 0;
    my $status;
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new();
    my $oUser = XDrive::DatabaseO::Table::UserData->new(undef, $oDiskAccount-
    >fetchDBO());

    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oUser->loadWhere("SEQ", $oDiskAccount->fetchColumn("USER_SEQ"));

    if ( (defined $oDiskAccount->fetchColumn("USER_SEQ"))
        && ($oUser->fetchColumn("EMAIL_ADDRESS") eq $sEmailAddress)
        )
    {
        my $sPassEncrypted = XDEncrypt($sPassword);
        $oDiskAccount->setColumn("PASSWORD", $sPassEncrypted);
        $oDiskAccount->update();
        $oDiskAccount->commit();
        $bReturnValue = 1;
    }
    elsif( (defined $oDiskAccount->fetchColumn("USER_SEQ"))
        && ($oUser->fetchColumn("EMAIL_ADDRESS") ne $sEmailAddress)
        )
    {
        &sendMail($oContent, $sUsername, "", $alert_template, " NOT");
    }

    $oDiskAccount->finish();
    $oDiskAccount->disconnect();

    return $bReturnValue;
}

sub display_form {
    my ($oContent, $template) = @_;
    $oContent->load($template);
}

sub sendMail {
    my ($oContent, $username, $password, $template, $not) = @_;

    my ($name_first, $name_last, $email_address, $data);
    my $oSearch = XDrive::DatabaseO::Search->new(undef);

```

```

$data = $oSearch->XDUserInfoByUsername($username);
$name_first = $data->[0]->[0];
$name_last = $data->[0]->[1];
$email_address = $data->[0]->[2];
$username = $data->[0]->[3];

my $message = &get_message($oContent,$name_first, $name_last, $username,
$password, $template);

my %toXdrive =
(
    To      => "$name_first $name_last <$email_address>",
    Bcc     => '',
    From    => "support@xdrive.com",
    Message => $message,
    Subject => "X:drive Password$not Updated!"
);

sendmail(%toXdrive);
)

sub get_message {
    my ($oContent,$name_first, $name_last, $username, $password, $template) =
    @_;

    $name_first = ($name_first)? $name_first : "";
    $name_last = ($name_last)? $name_last : "";

    $oContent->load($template);
    $oContent->tags
    (
        (
            'name_first' => $name_first,
            'name_last' => $name_last,
            'password' => $password,
            'username' => $username,
        )
    );

    return $oContent->get;
}

```

###forgot_username.cgi

```
#!/usr/bin/perl

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header param);
use CGI::Carp qw(fatalsToBrowser);
use Mail::Sendmail;

use Token;
# use XDrive::CGI qw(:MAIN);
use XDrive::Template;
use XDrive::DatabaseO::Search;
use XDrive::Library;
use XDrive::Utils::RandomString;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;

#####
my $invalid_template = "invalid_email.thtml";
my $request_template = "forgot_username_request.thtml";
my $thank_you_template = "forgot_username__t_y.thtml";
my $email_template = "forgot_username_email.thtml";
#####

exit &main();

sub main {
    my $oCGI = CGI->new();

    my $sEmailAddress = $oCGI->param('txtEmailAddress');
    my ($ar_usernames, $length);

    my $oSearch = XDrive::DatabaseO::Search->new(undef);

    my $oContent = new XDrive::Template;
    my $oNavigation = new XDrive::Template;
    my $oLayout = new XDrive::Template;

    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');
    $oLayout->partner('xdrv');

    ## Load the required template HTML files.
    $oNavigation->load("front_nav.thtml");
    $oLayout->load("layout.thtml");

    ## IF a parameter of email adress has been processed
    ## and in the correct format, then retrieve usernames
    ## associated with the email and send them.

    if ($sEmailAddress)
    {
        ## * * * * *
        ## added by kanlaya to check for correct email format
        ## * * * * *

        if ($sEmailAddress =~ /.*\@.*\./)

```



```

    {

        ## Takes the email_address and returns an array_ref
        ## of all the disk_account.usernames associated
        ## with that users.user.email_address
        $ar_usernames = $oSearch->XDUsernameFromEmail($sEmailAddress);
        $length = @$ar_usernames;

        ## IF there are usernames found for this address,
        ## then email the address the list of usernames.
        if($length > 0)
        {
            &sendMail($ar_usernames, $sEmailAddress, $length);
        }

        $oContent->load($thank_you_template);
        $oContent->tags({'emailAddress' => $sEmailAddress,});
    }

    else
        {$oContent->load($invalid_template);}
## * * * * *
## end add
## * * * * *
}
else
    {$oContent->load($request_template);}

    ## Print out the HTML and exit
    $oLayout->tags
    (
        'header_graphic' => 'header_fill.gif',
        'title' => 'What is my username?',
        'content' => $oContent->get,
        'navigation' => $oNavigation->get,
    );
    $oLayout->clear;

    print header, $oLayout->get;

    return 0;
}

sub sendMail {
    my ($usernames, $email, $length) = @_;

    my $message = &get_message($usernames, $email, $length);

    my %toXdrive =
    (
        To      => "$email",
        Bcc     => '',
        From    => "support\@xdrive.com",
        Message => $message,
        Subject => "X:drive Username Reminder"
    );

    sendmail(%toXdrive);
}

sub get_message {
    my ($usernames, $email, $length) = @_;

```

```
my ($sUsername, $sPlurals, $sPluralVerb);
$sUsername = join("\n", @$usernames);
$sPlurals = ($length > 1)? "s" : "";
$sPluralVerb = ($length > 1)? "are" : "is";

my $oForm = new XDrive::Template;
$oForm->partner('xdrv');

$oForm->load($email_template);
$oForm->tags
(
    'sEmailAddress' => $email,
    'sUsername' => $sUsername,
    'sPlurals' => $sPlurals,
    'sPluralVerb' => $sPluralVerb
);
$oForm->clear;

return $oForm->get;
}
```

###frame_generic.cgi

```

#!/usr/bin/perl
## Written by Matt Clapp on 6/28/99
## This CGI allows us to pass the sst and sid on to the inner frame

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use Token;
use XDrive::Library;
use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::Client::Security;
use XDrive::CGI qw(XDErrorToBrowser);
use XDrive::CGI::Cookie;

&main;
exit;

sub main
{
    my $oCGI = CGI->new();

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $language = $oCookie->getElement('language');
    $language = 'english' unless $language;

    my $sThtmlFile = $oCGI->param('thtml');
    my $sFrameHeight = $oCGI->param('sFrameHeight');

    if ($sFrameHeight == "")
    {
        $sFrameHeight="40";
    }

    if ($sThtmlFile eq 'download_client.thtml')
    {
        my $oTemplate = new XDrive::Template( { 'partner_code' => 'xdrv' } );
        $oTemplate->load($sThtmlFile);
        $oTemplate->tags( { 'sFrameHeight' => $sFrameHeight,
                           'language' => $language } );
        print "Content-type: text/html\n\n";
        print $oTemplate->get();
    }
    elsif ($sThtmlFile eq 'centerview.thtml')
    {
        my $sFrameSet;
        if ($sFrameHeight > 1)
        {
            $sFrameSet = "$sFrameHeight,*";
        }
        else
        {
            $sFrameSet = "100%,*";
        }
    }
};

```

```

        print <<EOM;
Content-type: text/html

```

```

<FRAMESET ROWS="$sFrameSet" BORDER=0 FRAMEBORDER=0 MARGINWIDTH=0
MARGINHEIGHT=0 TOPMARGIN=0 LEFTMARGIN=0 frameBorder=0 frameSpacing=0>
EOM

```

```

        if ($sFrameHeight > 1)
        {
            print <<EOM;
            <FRAME NAME='controls' SRC='/explorer/$language/buttons.html'
SCROLLING=NO MARGINWIDTH=0 MARGINHEIGHT=0 TOPMARGIN=0 LEFTMARGIN=0>
EOM

        }

        print <<EOM;
        <FRAME NAME='userData' SRC='/cgi-bin/explorer_user_data.cgi'
SCROLLING=AUTO MARGINWIDTH=0 MARGINHEIGHT=0 TOPMARGIN=0 LEFTMARGIN=0>
</FRAMESET>
EOM
    }
    else
    {
        ## Security check. Since the thtml file is passed in via the URL
the server ## can be hacked by passing in ../ offsets to get the directory
the hacker ## wants. A cleaner way would be to pass in a number and use
that number ## to access a hash, and die with a security violation if no such
has key ## exists.

        my $oDBO = XDrive::DatabaseO->new(undef,undef);
        my $oErr = new XDrive::Error;
        my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

        ####
        ## If the user failed to authenticate or an error occurred then
        ## redirect them to the error CGI and exit
        ####

        if ($oErr->Occurud)
        {
            xd_fatal_error($oCGI,$oErr);
            $oDBO->disconnect();
        }

        warn "##ALERT hacking attempt by ".$oToken->data('user').
            " from ".$ENV{REMOTE_IP};
        my $sMessage = $oErr->ReturnMessageGivenCode(341);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
        $oDBO->disconnect();
        exit;
    }
}

```

###get_a_shared_file.cgi

```

#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);

use XDrive::CGI;
use XDrive::Template;
use XDrive::DatabaseO::Table::DiskItemShare;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::Reseller;
use XDrive::CGI::Cookie;

use strict;

exit &main();

sub main {
    my $cgi = CGI->new();
    my ($ClaimTicket, $oPage, $xdDBH);

    if ($ENV{'QUERY_STRING'} !~ /=/)
    {
        $ClaimTicket = $ENV{'QUERY_STRING'};
    }
    else
    {
        $ClaimTicket = $cgi->param("claim_ticket");
    }

    if (length($ClaimTicket) < 5)
    {
        $ClaimTicket = $ENV{'PATH_INFO'};
        $ClaimTicket =~ s/^\///;
    }

    ##make sure that if claim ticket ends in -SP we set language to spanish
and
    ##truncate claim ticket
    if ($ClaimTicket =~ /-SP$/)
    {
        $ClaimTicket = substr($ClaimTicket,0,length($ClaimTicket)-3);
        my $oCookie = new XDrive::CGI::Cookie('x_session_info', $cgi);
        $oCookie->setElement
        (
            {
                'language' => 'spanish',
            }
        );

        print "Set-Cookie: ". $oCookie->asString();
    }

    my $oDiskItemShare = XDrive::DatabaseO::Table::DiskItemShare->new();
    $oDiskItemShare->loadWhere("RANDOM_KEY", $ClaimTicket);

    my $diskAccount = $oDiskItemShare->
    >fetchColumn("DISK_ACCOUNT_USER_SEQ");
    $xdDBH = $oDiskItemShare->fetchDBO();
}

```

```

my $oUserAccount = XDrive::DatabaseO::Table::UserData->new(undef,
$xddbH);
$oUserAccount->loadWhere("SEQ", $diskAccount);

my $oReseller = XDrive::DatabaseO::Table::Reseller->new(undef,
$xddbH);
$oReseller->loadWhere("SEQ", $oUserAccount-
>fetchColumn("RESELLER_SEQ"));

my $partner = $oReseller->fetchColumn("CODE");

## If the disk item share was not in the database then just use an
xdrive
## look n' feel. NOTE!!!! This should be changed to a plain looking
## error screen.
$partner = 'xdrv' if ! defined $partner;

$oPage = new XDrive::Template
(
    'partner_code' => $partner
);

$oPage->load('get_a_shared_file__frameset.thtml');
$oPage->tags
(
    'ClaimTicket' => $ClaimTicket,
    'referee' => $diskAccount,
);
$oPage->clear();
print $cgi->header, $oPage->get;

$oDiskItemShare->disconnect();
return 0;
}

```

###get_a_shared_file_download.cgi

```

#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use Data::Dumper;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::DatabaseO::Search;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::DiskItemShare;
use XDrive::Template;
use XDrive::Error;

use strict;

&main;
exit;

sub main
{
    my ($sFileDescription, $sFileSize, $sRandomKey, $sSeq);

    my $cgi = CGI->new();
    my $oErr = new XDrive::Error;
    my $g_oShared;    ## Shared object
    my $g_oSearch;    ## Shared object
    my $g_oAction;    ## Action object
    my $g_oFileStat; ## File stats
    $sRandomKey = $ENV{'QUERY_STRING'};

    if (!$sRandomKey)
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1360);
        &display_error($sMessage, $oErr);
    }
    else
    {
        ## Instantiate and load the shared object.
        $g_oShared = XDrive::DatabaseO::Table::DiskItemShare->new(undef,
undef);
        $g_oSearch = XDrive::DatabaseO::Search->new($g_oShared-
>fetchDBO());

        $g_oShared->loadWhere("RANDOM_KEY", $sRandomKey);
        $sSeq = $g_oShared->fetchColumn("SEQ");

        if (!$sSeq)
        {
            my $sMessage = $oErr->ReturnMessageGivenCode(1361);
            &display_error($sMessage, $oErr);
        }

        ## Call the client action constructor with the shared object
        ## which it will use to load all the needed client information.
        $g_oAction = new XDrive::Client::Actions($g_oShared, $cgi);

        my $sFile = join
            ('/',

```

```

        $g_oShared->fetchColumn("ITEM_PATH"),
        $g_oShared->fetchColumn("ITEM_NAME")
    );
    $g_oFileStat = $g_oAction->FileStat($sFile);

    if (!$g_oFileStat) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1362);
        &display_error($sMessage,$oErr);
    } else {

        $sFileDescription = $g_oShared->fetchColumn("DESCRIPTION");
        $sFileSize = ($g_oFileStat->size() > 1024)?
int($g_oFileStat->size()/1024) . "K" :

    $g_oFileStat->size() . " bytes";

        &display_form($g_oShared-
>fetchColumn("ITEM_NAME"),$sRandomKey, $sFileSize,
$sFileDescription,$g_oSearch->XDResellerCodeFromUserSeq($g_oShared-
>fetchColumn("DISK_ACCOUNT_USER_SEQ")));
    }
    $g_oShared->finish();
    $g_oShared->disconnect();
    $g_oAction->DisconnectDB();
}

sub display_form
{
    my ($sFilename,$sRandomKey, $sFileSize, $sFileDescription,$sPartner)
=@_;

    my $oForm = new XDrive::Template;
    $oForm->partner($sPartner);
    $oForm->load('get_a_shared_file__download_screen.shtml');
    $oForm->tags
    (
        {
            'sFilename' => $sFilename,
            'sExtraPathInfo' => $sFilename,
            'sRandomKey' => $sRandomKey,
            'sFileSize' => $sFileSize,
            'sFileDescription' => $sFileDescription,
        }
    );

    $oForm->clear();
    print "content-type: text/html\n\n", $oForm->get;
    exit(0);
}

sub display_error
{
    my ($message,$oErr) = @_;

    if (!$message)
    {
        $message = $oErr->ReturnMessageGivenCode(1363);
    }

    my $oForm = new XDrive::Template;
    $oForm->partner('xdrv');
    $oForm->load('get_a_shared_file__error.shtml');
}

```



```
$oForm->tags
((
    'message' => $message,
));
print "content-type: text/html\n\n", $oForm->get;
exit(0);
}
```

###login.cgi

```
#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> to verify that the user is
# good to login, if they are then log them in and otherwise redirect to
# a not authorized page.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::UserQuota;
use XDrive::DatabaseO::Table::Language;
use XDrive::DatabaseO::Search;
use CGI qw(param redirect header cookie);
use CGI;
use XDrive::CGI::Cookie;
use CGI::Carp qw(fatalsToBrowser);
```

```
use XDrive::CGI;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Template;
use XDrive::Library;
use XDrive::DatabaseO;
use Mail::Sendmail;
```

```
&main;
exit;
```

```
sub main
{
    my $oCGI      = new CGI;
    my $oErr      = new XDrive::Error;
    ##my $oDBO    = new XDrive::DatabaseO;
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    my $oToken;
    my $sToken;
    my $sUsername;
    my $sPartnerCode;
    ## johngaa add for dbexist check
    my $oDBO;

    if (XDDBConnectionCheck() && XDNFSCheck())
    {
        $oDBO = new XDrive::DatabaseO;
    }
    else
    {
        $oDBO = undef;
        print redirect("/upgrading_index.html");
        exit;
    }
    ## end of johngaa change
    my $bSecurity = $oCGI->param('bSecurity');
    my $sPartnerToken = $oCGI->param('partner_token');

    my $passed_lang = $oCGI->param('language');

    #####
```

```

## Attempt to authenticate the user by using one of the following two
## authentication methods: username/password pair or partner token
## authentication.
#####
if (! defined $sUsername && length($sPartnerToken) > 20)
{
    authPartnerUser($oCGI,$oErr,$oDBO,\$sUsername,\$oToken,
        \$sPartnerCode,$sPartnerToken);
    $sToken = $oToken->name();
}
else
{
    authWebSiteUser($oCGI,$oErr,$oDBO,\$sUsername,\$oToken);
    $sPartnerCode = 'xdrv';
}

#####
## If an error occurred while trying to create a token then redirect
## the user to the error page.
#####
if ($oErr->Occurred)
{
    $oDBO->disconnect;
    xd_fatal_error($oCGI,$oErr);
    exit;
}

#####
## If we have gotten here then we have an authenticated user.
#####

#####
## Build and print out cookies
#####
my $sLanguage = getLanguage($oDBO,$sUsername);

##check if user's language is the same as passed language
if ((length($passed_lang) > 0) && $sLanguage ne $passed_lang)
{
    ##update db here to new language
    setLanguage($oDBO,$sUsername,$passed_lang);
    ##update session to new language
    $sLanguage = $passed_lang;
}

##delete the promo cookie; this will not be set here and we
##don't want an old one hanging out
##promo cookies should be set in promo.cgi
$oCookie->deleteElement('promo') if $oCookie->getElement('promo');

$oCookie->setElement
(
    (
        'language' => $sLanguage,
        'partner' => $sPartnerCode,
    )
);

print "Set-Cookie: ". $oCookie->asString();
print "Set-Cookie: SST=$sToken; domain=.xdrive.com; path=/" . "\n"
    if $sPartnerCode ne 'xdrv';

```

```

#####
## write user login to the database
#####
&incrementLoginNumber($oDBO,$sUsername,$sLanguage,$sPartnerCode);

#####
## Send the user off into thier file explorer
#####
if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
{
    print $oCGI->redirect("?sst=".$oToken->name()."&sid=0");
}
else
{
    xd_web_open($oCGI, "", "", \%ENV, $bSecurity);
}

$oDBO->disconnect;
return 0;
}

```

```

sub isYesterday()
{
##
## Date: 01/25/99
## used to check of a date if its today or not
##
    my $last_login = `shift`;
    my $nSec;           ## Seconds
    my $nMin;          ## Minutes
    my $nHour;         ## Hours
    my $sDay;          ## Weekday
    my $nDay;          ## Numeric date (01-31)
    my $nMonth;        ## Numeric month (01-12)
    my $nYear;         ## Numeric year (00-99)

    my $todaysDate = ($nSec, $nMin, $nHour, $nDay, $nMonth, $nYear,
    $sDay) = (localtime(time))[0,1,2,3,4,5,6];

    $last_login =~ /([\d]+)-([\d]+)-([\d]+)/i;
    my $last_login_year = int($1);
    my $last_login_month = int($2);
    my $last_login_day = int($3);

    if ($last_login_year < $nYear)
    {
        return 1;
    }
    if ($last_login_month < $nMonth)
    {
        return 1;
    }
    if ($last_login_day < $nDay)
    {
        return 1;
    }
    return 0;
}

```

```

sub incrementLoginNumber()
{
    my $oDBO = shift;
    my $sUsername = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $timesLoggedIn = $oDiskAccount->fetchColumn("LOGIN_NUM");
    my $user_seq = $oDiskAccount->fetchColumn("USER_SEQ");

    ## johngaa add
    ## insert a warn in the error log if this is the
    if ($ENV{'HTTP_USER_AGENT'} =~ /^xdwin/)
    {
        my $todaysDate = XDToday();
        warn "#client_login user_seq=$user_seq username=$sUsername
date=$todaysDate#";
    }
    ## end of johngaa warn of first entry

    if ($timesLoggedIn)
    {
        $timesLoggedIn++;
    }
    else
    {
        $timesLoggedIn=1;
    }

    $oDiskAccount->setColumn("LOGIN_NUM", $timesLoggedIn);
    $oDiskAccount->setColumn("LAST_LOGIN",XDToday());

    my $status = $oDiskAccount->update();

    if ($status > -1)
    {
        $oDiskAccount->commit();
        $oDiskAccount->finish();

        ## johngaa modify to exclude college club
        ## and quepasa users out of the extra space
        ## promo

        my $oUserData = XDrive::DatabaseO::Table::UserData-
>new(undef,$oDBO);
        $oUserData->loadWhere("SEQ", $user_seq);
        my $reseller_seq = $oUserData-
>fetchColumn("RESELLER_SEQ");
        if (!(isResellerSeqCC_QUPA($oDBO,$reseller_seq)))
        {
            ##give user extra 10MB if 10th login
            if ($timesLoggedIn == 10)
            {
                my $oUserQuota = XDrive::DatabaseO::Table::UserQuota-
>new(undef,$oDBO);

```

```

        $oUserQuota->loadWhere("USER_SEQ", $user_seq);
        my $additional_quota = $oUserQuota-
>incrementQuota($user_seq,10240);
        if ($additional_quota > 0)
        {
            &send_email($user_seq, $oDBO,
$additional_quota,$sLanguage, $sPartnerCode);
        }
    }
}
else
{
    # $oDiskAccount->rollback();
}
}

```

```

sub isResellerSeqCC_QUPA
{

```

```

    my $oDBO = shift;
    my $reseller_seq = shift;
    my $dbh = $oDBO->fetchDBH();

    my $sql_stmt = "SELECT code FROM reseller WHERE seq=?";
    my $cmd;
    my @data;

    $cmd = $dbh->prepare($sql_stmt);
    $cmd->execute(($reseller_seq));
    @data = $cmd->fetchrow_array;
    if ($data[0] eq 'cc' || $data[0] eq 'qupa')
    {
        return 1;
        ##print "should return a true\n"
    }
    return 0;
}

```

```

sub send_email
{

```

```

    my $user_seq = shift;
    my $oDBO = shift;
    my $additional_quota = shift;
    my $sLanguage = shift;
    my $sPartnerCode = shift;

    if ($sPartnerCode eq 'cc')
    {
        return;
    }
}

```

```

##comes in as k, change to megabytes
my $mbs = $additional_quota/1024;

```

```

    my $oUserData = XDrive::Database0::Table::UserData->new(undef,$oDBO);
    $oUserData->loadWhere("SEQ", $user_seq);
    my $email_address = $oUserData->fetchColumn("EMAIL_ADDRESS");
    my $name_first = $oUserData->fetchColumn("NAME_FIRST");
    my $name_last = $oUserData->fetchColumn("NAME_LAST");

```

```

my $oTemplate = new XDrive::Template
({
  'language'      => $sLanguage,
  'partner_code' => $sPartnerCode,
});

$oTemplate->load('received_10MB_10logins.thtml');
$oTemplate->tags({
  'mbs' => $mbs,
});
$oTemplate->clear();
my $message = $oTemplate->get;

my %toXdrive =
(
  To      => "$name_first $name_last <$email_address>",
  Bcc     => '',
  From    => "support@xdrive.com",
  Message => $message,
  Subject => "Congratulations!"
);

sendmail(%toXdrive);
)

sub authPartnerUser
{
  my $oCGI = shift;
  my $oErr = shift;
  my $oDBO = shift;
  my $rsUsername = shift;
  my $roToken = shift;
  my $rsPartnerCode = shift;
  my $sPartnerToken = shift;

  my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);
  my $oPartnerToken = new Token
  (
    {
      'err' => $oErr,
      'dbh' => $oDBO,
    }
  );
  $oPartnerToken->load($sPartnerToken);

  return if $oErr->Occurud;

  $$roToken = new Token
  (
    {
      'dbh' => $oDBO,
      'err' => $oErr,
      'user_sequence' => $oPartnerToken->data('user_seq'),
    }
  );
  $$roToken->create();

  return if $oErr->Occurud;

  ### Edited by Justin so that the partner_code is looked for
  ### in the cookie instead of the token table.
  ### And then again because I shouldn't have done that. The
  ### partner code hasn't been set in the cookie by this point,
  ### so we shouldn't be looking in there for it.
  $rsPartnerCode = $oPartnerToken->data('partner_code');
  # $rsPartnerCode = $oCookie->getElement('partner');

```

```

    $$rsUsername = $oPartnerToken->data('user');

    $$roToken->data('ip', $ENV{REMOTE_ADDR});
    $$roToken->data('browser', $ENV{HTTP_USER_AGENT});
    $$roToken->data('user', $$rsUsername);
    $$roToken->data('user_seq', $oPartnerToken->data('user_seq'));
    $$roToken->data('partner_code', $$rsPartnerCode);
    $$roToken->data('disk_account_seq', $oPartnerToken-
>data('disk_account_seq'));
    $$roToken->save;

    $oPartnerToken->delete();
}

sub authWebSiteUser
{
    my $oCGI = shift;
    my $oErr = shift;
    my $oDBO = shift;
    my $rsUsername = shift;
    my $roToken = shift;

    my $sPassword = $oCGI->param('pass');
    $$rsUsername = $oCGI->param('user');

    if (xd_auth_password($$rsUsername, $sPassword, $oDBO))
    {
        ## Login the user info X:drive and get the session token
        $$roToken = xd_login($oCGI, $$rsUsername, $oErr, $oDBO);
    }
    else
    {
        $oErr->AddErrorByErrorCode('501');
    }
}

sub getLanguage
{
    my $oDBO = shift;
    my $sUsername = shift;

    my $language;

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef, $oDBO);
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings-
>new(undef, $oDBO);
    $oUserSettings->loadWhere("USER_SEQ", $userSeq);
    $oUserSettings->finish;
    my $language = $oUserSettings->fetchColumn("LANGUAGE");

    if ($language eq '')
    {
        $language = 'english';
    }
    else
    {
        ## Get language from database given code

```



```

        my $oLanguage = XDrive::DatabaseO::Table::Language-
>new(undef,$oDBO);
        $oLanguage->loadWhere("SEQ",$language);
        $oLanguage->finish;
        $language = $oLanguage->fetchColumn("CODE");
    }

    return $language;
}

sub setLanguage
{
    ##set the LANGUAGE column of the User_Settings table to passed
    language

    my $oDBO = shift;
    my $sUsername = shift;
    my $language = shift;

    my ($rv,$errorCode);

    ## get the user's language out of the database
    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBO);
    ##grab right table
    $oDiskAccount->loadWhere("USERNAME", $sUsername);
    $oDiskAccount->finish;
    my $userSeq = $oDiskAccount->fetchColumn("USER_SEQ");

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings-
>new(undef,$oDBO);
    $oUserSettings->loadWhere("USER_SEQ",$userSeq);
    $oUserSettings->finish;

    ##grab the seq number of the LANGUAGE being passed
    my $oLanguage = XDrive::DatabaseO::Table::Language->new(undef,$oDBO);
    $oLanguage->loadWhere("CODE",$language);
    $oLanguage->finish();
    my $seq_lang = $oLanguage->fetchColumn("SEQ");

    eval
    {
        ##
        ##set language here
        $rv = 0;
        $oUserSettings->setColumn('LANGUAGE',$seq_lang);
        $rv = $oUserSettings->update();
    };
    if ($rv == 0)
    {
        $oUserSettings->rollback();
        $errorCode = 0;
    }
    else
    {
        $oUserSettings->commit();
        $errorCode = 1;
    }
    return $errorCode;
}

```

###logout.cgi

```
#!/usr/bin/perl
## Program to log the user out, currently hacked to redirect to the homepage.
## Modified by Justin on 10/15/99 to be Security.pm friendly
## and get rid of the XDrive::CGI stuff.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::Error;

&main;
exit;

sub main
{
    my $oCGI    = CGI->new();
    my $oDBO    = new XDrive::DatabaseO;
    my $oError  = new XDrive::Error;

    ##removes token from the database
    xd_logout($oDBO, $oCGI, $oError);

    $oDBO->disconnect;

    print $oCGI->redirect('/');
    return 0;
}
```

###navbar.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <martin@xdrive.com> on Sun Sep 5 1999
## Script to dynamically show the correct tempate based on which
## partner is looking at the web site.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use XDrive::Library;
use XDrive::Template;
use XDrive::Error;
use XDrive::DatabaseO;
use XDrive::Client::Security;

&main;
exit;

sub main
{
    ## Load the session token
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;
    my $oCGI = new CGI;
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    #####
    ## Attempt to autenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If the user does not validate or an error occurud
    ## then redirect to the error CGI and exit
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);

        $oDBO->disconnect();

        exit;
    }

    #####
    ## Otherwise we have validated and should load the navbar
    ## associated with the partner
    #####
    ### Edited by Justin so that partner_code is looked for in
    ### the cookie instead of the token table.
    # my $oForm = new XDrive::Template
    #     (
    #         'partner_code' => $oToken->data('partner_code')
    #     );

    my $oForm = new XDrive::Template
    (

```

```
'partner_code' => $oCookie->getElement('partner')
));

$oForm->load('navbar.shtml');

####
## Print the navbar and stop
####

print "Content-type: text/html\n\n";
print $oForm->get;

$oDBO->disconnect();

return 0;
}
```

###password_change.cgi

```

#!/usr/bin/perl
## Written by Lucas McGregor on ???

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header param);
use CGI::Carp qw(fatalsToBrowser);

use Token;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use XDrive::Client::Registration;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::Library;
use XDrive::Template;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;

&main;
exit;

sub main
{
    my $oCGI    = CGI->new();
    my $oDBO    = new XDrive::DatabaseO;
    my $oErr    = new XDrive::Error;

    ####
    ## Attempt to authenticate the user
    ####

    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    ####
    ## If an error occurs or the user fails to authenticate then redirect
    ## to the error CGI and exit
    ####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    ####
    ## Otherwise have have authenticated the user and can proceed
    ####

    my $sUsername = $oToken->data('user');

    my $sPasswordNew      = $oCGI->param('txtPasswordNew1');
    my $sPasswordNewConfirm = $oCGI->param('txtPasswordNew2');
    my $sPasswordOld      = $oCGI->param('txtPasswordOld1');

```

```

    if (($sPasswordNew eq '') || ($sPasswordNewConfirm eq '') ||
($sPasswordOld eq ''))
    {
        ##if any of the fields is blank, give em error message
        my $sMessage = $oErr->ReturnMessageGivenCode(1340);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    ## Change user's password
    PasswordSet($sUsername, $sPasswordNew,
$sPasswordOld, $oToken, $oErr, $oCGI);

    return 0;
}

```

```

#####
## PasswordSet: Change user's password
#####

```

```

sub PasswordSet($$)
{
    my $sUsername = shift;          ## (I) User in question
    my $sPassword = shift;          ## (I) New password
    my $sPasswordOld = shift;        ## (I) Old password
    my $oToken = shift;              ## (I) Token object
    my $oErr = shift;
    my $oCGI = shift;
    my $sPassEncrypted = XDEncrypt($sPassword);

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new();
    $oDiskAccount->loadWhere("USERNAME", $sUsername);

    if (! PasswordsMatch($oDiskAccount-
>fetchColumn("PASSWORD"), $sPasswordOld))
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1341);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    if (! defined $oDiskAccount->fetchColumn("USER_SEQ"))
    {
        my $sMessage = $oErr->ReturnMessageGivenCode(1342);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    $oDiskAccount->setColumn("PASSWORD", $sPassEncrypted);
    my $status = $oDiskAccount->update();

    ## If no error, then commit
    ## Else rollback and show an error
    if ($status > -1) {
        $oDiskAccount->commit();
    }
    else
    {
        $oDiskAccount->rollback();
        my $sMessage = $oErr->ReturnMessageGivenCode(1343);
        XDErrorToBrowser("", $sMessage, undef, $oToken);
    }

    $oDiskAccount->finish();
    $oDiskAccount->disconnect();
}

```

```

my $oTemplate = new XDrive::Template( {'partner_code' => 'xdrv'} );
$oTemplate->load('password_changed.thtml');
print $oCGI->header(), $oTemplate->get;
}

```

```

#####
## PasswordsMatch: Check an encrypted password against an unencrypted
## password and return true or false.
#####

```

```

sub PasswordsMatch

```

```

{
    my $sEncrypted = shift; ## current password
    my $sToCheck   = shift; ## string to check

    ## Encrypt the passed password with the salt from the password taken
    ## from the database.
    my ($sSalt) = $sEncrypted =~ /^(\w{2})/;

    ## Do the passwords match? If so then return true, otherwise false.
    if ($sEncrypted eq crypt($sToCheck,$sSalt))
    {
        return TRUE;
    }

    return FALSE;
}

```

###promo.cgi

```

#!/usr/bin/perl
##
## File: promo.cgi
##
## Written by Justin White on 10/25/99.
## Sets a promo cookie and redirects to the home page.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::CGI::Cookie;
use XDrive::DatabaseO::Search;

use CGI;
use CGI::Carp qw(fatalsToBrowser);

&main();

exit;

sub main {
    my ($cookie, $promo, %new_info, $oSearch, $oTemplate);

    my $oCGI    = CGI->new();
    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    my $path_info = $ENV{'PATH_INFO'};

    my $sClaimTicket = $oCGI->param('ct');

    if ($sClaimTicket) {
        ##
        # Via cookie, set the promo so that signup_account.cgi treats
        # it as a promo and set the claim ticket code so that we can
        # remove that data from the batch_user_data table.
        ##
        $oCookie->setElement( {'ct' => $sClaimTicket} );
    }

    if ($path_info) {
        $path_info =~ s/^\///;

        $oCookie->setElement( {'promo' => $path_info} );
        $oCookie->setPath('/');

        ##if user is coming from the befree promo, set a cookie with their
        ##source id, be Free requires this for tracking purposes
        if ($path_info =~ /befree/)
        {
            my $sourceid = $oCGI->param('sourceid');
            print "Set-Cookie: sourceid=$sourceid; domain=.xdrive.com;
path=/\n"
        }

        my $oDBO    = XDrive::DatabaseO->new();
        my $oSearch = XDrive::DatabaseO::Search->new($oDBO);

```



```

my @bind_array = ($path_info);

# my $st = "SELECT p.template, p.redirect_url, dl.code
#         FROM xdrive.promo p, xdrive.v_language dl
#         WHERE p.uri = '$path_info'
#         AND p.du_language = dl.seq(+)";

my $st = "SELECT p.template, p.redirect_url, dl.code
        FROM xdrive.promo p, xdrive.v_language dl
        WHERE p.uri = ?
        AND p.du_language = dl.seq(+)";

# my $data = $oSearch->XDSQLSearch($st);
my $data = $oSearch->XDSQLSearch($st, \@bind_array);
my $rows = @{$data};

if ($rows > 0) {
    my $template      = $$data[0][0];
    my $redirect_url  = $$data[0][1];
    my $language      = $$data[0][3];

    $oCookie->setElement( {'language' => $language} );

    print "Set-Cookie: ", $oCookie->asString();

    if ($template) {
        eval {
            $oTemplate = new XDrive::Template( {'cookie'      =>
$SoCookie,
                                                'partner_code' => 'xdrv'
});

            $oTemplate->partner('xdrv');

            $oTemplate->load("promo/$template");
        };

        if ($?) {
            print $oCGI->redirect('/');
            warn "$?\n";
        }
        else {
            print $oCGI->header(), $oTemplate->get;
        }

        $oSearch->disconnect;
    }
    elsif ($redirect_url) {
        print $oCGI->redirect($redirect_url);

        $oSearch->disconnect;
    }
    else {
        print $oCGI->redirect('/');

        $oSearch->disconnect;
    }
}
else {
    print $oCGI->redirect('/');
}
}
else {

```

```
    print $oCGI->redirect('/');  
    $oSearch->disconnect;  
}  
return;  
)
```

###removespace.cgi

```

#!/usr/bin/perl
#####3
## Written by Karen Eppinger
## removespace.cgi - cancels additional space requests
#####3

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Error;
use XDrive::Library;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::Reseller;
use XDrive::DatabaseO::Table::Deal;
use XDrive::DatabaseO::Table::Item;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Table::UserPurchase;
use XDrive::Client::Actions;
use XDrive::DatabaseO::Search;
use XDrive::Sale::Purchase;
use Mail::Sendmail;
use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Template;
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Security;
use EpochClient_ssl;

use strict;

$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)); # Make %ENV safer

exit &main;

#####
## main: main function calls all others
##
##
#####

sub main
{
    my $oCGI = CGI->new();
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    ####
    ## Attempt to authenticate the user
    ####

    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    ####
    ## If an error occurs during authentication or authentication fails
    ## then redirect to the error CGI and exit
    ####

```

```

if ($oErr->Occurud)
{
    xd_fatal_error($oCGI,$oErr);
    exit;
}

####
## Otherwise we have a valid session
####

my $sUserName = $oToken->data('user');
my $oTemplate = new XDrive::Template
(
    'partner_code' => $oToken->data('partner_code')
);

## used to figure whether to give user the form or process the form
my $sAction = $oCGI->param("action");

## Create a DBH
my $oDBH = XDrive::DatabaseO->new();

## if the action is a request type
if ($sAction eq 'process')
{
    ##else we process the form input
    &CheckSpaceUsed($oCGI,$sUserName,$oTemplate,$oToken,$oDBH,$oErr);
}
elsif ($sAction eq 'intro')
{
    &ShowIntroPage($oTemplate,$sUserName,$oToken,$oCGI);
}
else
{
    ## we give the user the form
    &ShowSpace($sUserName,$oTemplate,$oToken,$oDBH,$oErr);
}
$oDBH->disconnect();
}

#####
## CheckSpaceUsed: make sure the user has enough free space for his files
## if not, do not let him cancel
#####

sub CheckSpaceUsed
{
    my $oCGI = shift;
    my $sUserName = shift;
    my $oTemplate = shift;
    my $oToken = shift;
    my $oDBH = shift;
    my $oErr = shift;

    ##we need to get the number of fields so we know what to process
    my @fields = $oCGI->param;
    my $checked = 0;

    my $returnValue = '';
    ##for each checked item, either cancel or tell user they may not cancel
    ##because space used is larger than space available after cancelation

```

```

for (my $i=0; $i<$#fields; $i++)
{
    if ($fields[$i] =~ /^tc_/)
    {
        $fields[$i] =~ s/^tc_//;
        my $oPurchase = new XDrive::Sale::Purchase($oDBH);
        my @message_dbmessage = $oPurchase->CancelItem($fields[$i],
$sUserName);
        $returnValue .= $message_dbmessage[0];
        $checked++;

        if ($message_dbmessage[1] != 0)
        {
            $oDBH->commit();
        }
        else
        {
            $oDBH->rollback();
        }
    }
}

if ($checked>0)
{
    ##show the page that tells user if space was cancelled or not
    &ShowCanceled($returnValue,$oTemplate);
}
else
{
    ##user hasn't checked anything, give em error page
    my $sError = $oErr->ReturnMessageGivenCode(1301);
    XDErrorToBrowser("", $sError, undef, $oToken);
}
}

```

```

#####
## ShowCanceled: tell user space was cancelled
#####

```

```

sub ShowCanceled
{
    my $sItemsCanceled = shift;
    my $oTemplate       = shift;

    ## Load the required template HTML files.
    $oTemplate->load('removespace_ok.thtml');
    $oTemplate->tags
    (
        {
            'items' => $sItemsCanceled
        }
    );
    print "Content-type: text/html\n\n";
    print $oTemplate->get();
}

```

```

#####
## ShowSpace: shows the user the initial page with their current space
## allocation
#####

```

```

sub ShowSpace
{

```

```

    my $sUserName = shift;
    my $oTemplate = shift;
    my $oToken = shift;
    my $oDBH = shift;
    my $oErr = shift;

    my $sMessage = $oErr->ReturnMessageGivenCode(1302);
    $sMessage = &GetItems($sUserName,$oToken,$oDBH,$oErr);

    ## Load the required template HTML files.
    $oTemplate->load('removespace_request.shtml');
    $oTemplate->tags
        (
            (
                'items' => $sMessage
            )
        );

    print "Content-type: text/html\n\n";
    print $oTemplate->get();
}

sub ShowIntroPage
{
    my $oTemplate = shift;
    my $sUserName = shift;
    my $oToken = shift;
    my $oCGI = shift;

    my $oAction = new XDrive::Client::Actions
        (
            $oToken,
            $oCGI
        );
    my $quotaUsed = $oAction->QuotaUsed();
    $quotaUsed = sprintf("%2.2f",$quotaUsed/1024);
    my $quotaLimit = $oAction->QuotaLimit();
    $quotaLimit = sprintf("%2.2f",$quotaLimit/1024);

    $oTemplate->load('removespace_intro.shtml');
    $oTemplate->tags
        (
            (
                'quotaUsed' => $quotaUsed,
                'quotaLimit' => $quotaLimit
            )
        );
    $oTemplate->clear();

    print "Content-type: text/html\n\n";
    print $oTemplate->get();
}

sub GetItems
{
    my $sUserName = shift;
    my $oToken = shift;
    my $oDBH = shift;
    my $oErr = shift;

    my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount-
>new(undef,$oDBH);
    $oDiskAccount->loadWhere('USERNAME', $sUserName);

    ##now load all items in the user_purchase database that are
    ##owned by this user
    my $userSeq = $oDiskAccount->fetchColumn('USER_SEQ');

```

```

##passing a 0 as the last parameter returns all non-canceled items
my $oSearch = XDrive::DatabaseO::Search->new(undef);
my $array = $oSearch->XDUserPurchases($userSeq, 0);

##see if the array returned any items

if ($array->[0][0] eq '')
{
    my $sError = $oErr->ReturnMessageGivenCode(1302);
    XDErrorToBrowser('removespace_noitems.shtml', $sError, 1,
$oToken);
}

my $i;
my $items = '';

for $i(0..$#($array))
{
    ##storing the complete string returned by Epoch
    ##must take only stuff after the | to cancel transaction
    ##and chop off last character which seems to be a line return
    ##may have to alter this if we see problems
    chop($array->[$i][4]);
    my @aCodes=split(/\|/, $array->[$i][4]);
    my $itemName = 'tc_' . $aCodes[1];
    $itemName=~s/~///;

    ##Get the name associated with this item
    my $oDeal = XDrive::DatabaseO::Table::Deal->new(undef,$oDBH);
    $oDeal->loadWhere('SEQ',$array->[$i][2]);
    my $itemSeq = $oDeal->fetchColumn('ITEM_SEQ');
    my $oItem = XDrive::DatabaseO::Table::Item->new(undef,$oDBH);
    $oItem->loadWhere('SEQ', $itemSeq);
    my $description = $oItem->fetchColumn('DESCRIPTION');

    $items .= '<input type="checkbox" name="' . $itemName . '">' .
$description . '<BR>';
}

if ($items eq '')
{
    my $sError = $oErr->ReturnMessageGivenCode(1302);
    XDErrorToBrowser('removespace_noitems.shtml', $sError, 1,
$oToken);
}

return $items;
}

```

###selected_delete.cgi

```

#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> for renaming files from the
# web.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::Client::Security;

use XDrive::Error;
use XDrive::DatabaseO;

exit &main;

sub main
{
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    #####
    ## Attempt to authenticate the user
    #####

    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);

    #####
    ## If an error occurred or the user could not be validated then
    ## redirect to the error CGI and exit
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI, $oErr);
        exit;
    }

    #####
    ## Otherwise we know that we have a valid session
    #####

    my $oAction = new XDrive::Client::Actions
    (
        $oToken,
        $oCGI
    );

    $oAction->FileCheck($oAction->ItemCurrent());
    $oAction->ItemDelete($oAction->ItemCurrent());
    xd_web_buttonindex($oCGI);
    $oAction->DisconnectDB();

    return 0;
}

```


###selected_rename.cgi

```

#!/usr/bin/perl
# Written by Martin Hald <mhald@geotribe.com> for renaming files from the
# web.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp 'fatalsToBrowser';
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Actions;
use XDrive::Client::Security;

use XDrive::Library;
use XDrive::DatabaseO;
use XDrive::Error;

## Clean up the path
$ENV{'PATH'} = '/bin';
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)); # Make %ENV safer

exit &main;

sub main {
    my $oCGI = new CGI;
    my $oErr = new XDrive::Error;
    my $oDBO = new XDrive::DatabaseO;

    ####
    ## Attempt to authenticate the user
    ####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    ####
    ## If the authentication fails or there is an error during the
    ## authentication phase then redirect to the error CGI
    ####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    ####
    ## Otherwise we have a valid session
    ####

    my $oAction = new XDrive::Client::Actions
    (
        $oToken,
        $oCGI
    );

    my $sItemOld = $oAction->ItemCurrent();

    ## Get the relative path to the item to be renamed from the
    ## old item name itself.

```

```
my ($sFolder) = $sItemOld =~ /(.\+\/){^\+\/}+//;

## Set the new item to be in that folder.
my $sItemNew = $sFolder.$oAction->ItemNew().$oAction-
>ItemExtension();

$oAction->FileCheck($sItemOld);
$oAction->ItemRename($sItemOld,$sItemNew);

xd_web_buttonindex($oCGI);
$oAction->DisconnectDB();
)
```

###settings_save.cgi

```

#!/usr/bin/perl

use strict;
use vars qw(@ISA);
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use Data::Dumper;
use XDrive::Library;
use XDrive::CGI;
use XDrive::Client::Quota;
use XDrive::Client::Security;
use XDrive::CGI::Cookie;
use XDrive::DatabaseO::Table::UserSettings;
use XDrive::DatabaseO::Table::Language;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::Template;

@ISA = qw(XDrive::CGI);

exit &main;

sub main {
    my $oCGI = CGI->new();
    my $oDBO = new XDrive::DatabaseO;
    my $oErr = new XDrive::Error;

    ####
    ## Attempt to autenticate the user
    ####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    ####
    ## If the autentication fails or there is an error during the
    ## autentication phase then redirect to the error CGI
    ####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    ####
    ## Otherwise we have a valid session
    ####

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $sUser = $oToken->data('user');
    my $nUser = UserIdGet($sUser);

    my $oUserSettings = XDrive::DatabaseO::Table::UserSettings->new(undef,
undef);

    ## Initialize global variables
    my $g_bFileExtEdit = $oCGI->param('bFileExtEdit') eq 'on' ? 1 : 0;

```

```

my $g_bExtraHelp = $oCGI->param('bExtraHelp') eq 'on' ? 1 : 0;
my $g_bMarketing = $oCGI->param('bMarketing') eq 'on' ? 1 : 0;
my $g_bNewsletter = $oCGI->param('bNewsletter') eq 'on' ? 1 : 0;
my $g_bLanguage = $oCGI->param('bLanguage');
my $sCurrentLanguage;

my $languageCode;

if (defined $g_bLanguage)
{
    my $oLanguage = XDrive::DatabaseO::Table::Language->new
        (undef, $oUserSettings->fetchDBO());
    $oLanguage->loadWhere("CODE", $g_bLanguage);
    $languageCode = $oLanguage->fetchColumn("SEQ");
}

## We are doing this in a backwards way -- first we will try and load
the ## current users profile. If that works then we change it and update
it ## by calling save. If that does not work then we just call save.

$oUserSettings->loadWhere("USER_SEQ", $nUser);
$oUserSettings->setColumn("FILE_EXT_EDITABLE", $g_bFileExtEdit);
$oUserSettings->setColumn("EXTRA_HELP", $g_bExtraHelp);
$oUserSettings->setColumn("OPT_MARKETING", $g_bMarketing);
$oUserSettings->setColumn("OPT_NEWSLETTER", $g_bNewsletter);

## The language element is an OPTIONAL setting in the "My Profile"
area. ## If it is passed then set it, otherwise leave the current value.
if (defined $g_bLanguage)
{
    $sCurrentLanguage = $g_bLanguage;
    $oUserSettings->setColumn("LANGUAGE", $languageCode);
}
else
{
    $sCurrentLanguage = "english";
}

my $status = $oUserSettings->update();

if ($status < 0)
{
    $oUserSettings->rollback();
    my $sMessage = $oErr->ReturnMessageGivenCode(1330);
    XDErrorToBrowser(undef, $sMessage, undef, $oToken)
}
else
{
    $oUserSettings->commit();

    if (defined $g_bLanguage)
    {
        ##set the cookie for language
        $oCookie->setElement
        (
            {
                'language' => $g_bLanguage
            }
        );
        print "Set-Cookie: ", $oCookie->asString();
    }
}

```

```
## Redirect the browser to the succesfull save page.

### Edited by Justin so that we get the partner_code out
### of cookie instead of the token table.
# print xd_web_redirect
# (
#     "/account/profile/$sCurrentLanguage/saved.html",
#     $oToken->data('partner_code')
# );

print xd_web_redirect
(
    "/account/profile/$sCurrentLanguage/saved.html",
    $oCookie->getElement('partner')
);

$oUserSettings->finish();
$oUserSettings->disconnect();
$oDBO->disconnect();
}
```

###share_a_file.cgi

```
#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use XDrive::Client::Quota;
use Math::TrulyRandom;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Search;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::UserData;
use XDrive::Utils::RandomString;
use XDrive::CGI;

use Mail::Sendmail;
use CGI::Carp qw(fatalsToBrowser);
use CGI;
use XDrive::Template;
use XDrive::Client::Security;
use XDrive::Error;
use XDrive::Library;
use XDrive::CGI::Cookie;

use strict;

&main();

sub main {
    my $cgi = CGI->new();
    my $oErr = new XDrive::Error;
    my $xdDBH = XDrive::DatabaseO->new();

    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $cgi);

    #####
    ## Attempt to authenticate the user
    #####

    my $oToken = xd_security_check($xdDBH,$cgi,$oErr);

    #####
    ## If the authentication fails or there is an error during the
    ## authentication phase then redirect to the error CGI
    #####

    if ($oErr->Occurud) {
        xd_fatal_error($cgi,$oErr);
        $xdDBH->disconnect();
        exit;
    }

    #####
    ## Otherwise we have a valid session
    #####

    ### Edited by Justin so that the partner_code is looked for
    ### in the cookie instead of the token table.
    # my $sPartner = $oToken->data('partner_code');
    my $sPartner = $oCookie->getElement('partner');
    my $nUser_ID = UserIdGet($oToken->data('user'));

```

```

## Grab the user info from the Database
my $oUserInfo = XDrive::DatabaseO::Table::UserData->new({}, $xdDBH);

my $sFileName = $cgi->param("sFileName");
my $bHelp = $cgi->param("help");

my $sFriendsEmail = &get_friends_emails($cgi);
my $sEmailSubject = $cgi->param('sEmailSubject');
my $sEmailMessage = $cgi->param("sEmailMessage");
my $sFileDescription = $cgi->param("sFileDescription");

my ($sRandomKey, $sFilePath);

## Load user info where the SEQ = $nUser_ID
$oUserInfo->loadWhere("SEQ", $nUser_ID);

my $sUser_name = $oUserInfo->fetchColumn("NAME_FIRST") . " " .
$oUserInfo->fetchColumn("NAME_LAST");
my $sUser_email = $oUserInfo->fetchColumn("EMAIL_ADDRESS");

if ($sFriendsEmail)
{
    $sFilePath="/";
    $sFileName =~ m%(.*)/(.*)%;

    #inserted this code to catch documents that are not in a folder

    my $tempFilePath = "/" . $1;
    my $tempFileName = $2;

    if ($tempFileName ne "")
    {
        $sFileName=$tempFileName;
        $sFilePath=$tempFilePath;
    }

    &verify_database_values($nUser_ID, $sFileName, $sFilePath,
        $sFilePath, $sFileName,
        $sFileDescription,$oToken,$oErr);

    ## Insert the info into the disk_item_share table, and get the
    random key
    $sRandomKey = &insert_file_into_database($nUser_ID, $sFileName,
        $sFilePath, $sFileDescription, $xdDBH,$oToken,$oErr);

    &send_mail($sFriendsEmail, $sEmailSubject, $sEmailMessage, $sFileDescription,
        $sUser_name, $sUser_email, $nUser_ID,
        $sRandomKey,$sPartner,$oToken,$oErr,$cgi);

    &display_thank_you($sPartner);
}
else {
    $oUserInfo->finish();
    $xdDBH->disconnect();
    &display_form($sFileName, $bHelp, $sPartner);
}

$oUserInfo->finish();
$oUserInfo->disconnect();
}

```

```

sub send_mail {
    my ($sFriendsEmail, $sEmailSubject, $sEmailMessage, $sFileDescription,
        $sUser_name, $sUser_email, $nUser_ID, $sRandomKey,
        $sPartner,$oToken,$oErr,$oCGI) = @_;

    ##get language from the cookie.  If not english, append language code
to url
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);
    my $language = $oCookie->getElement('language');
    if ($language ne 'english')
    {
        if ($language eq 'spanish')
        {
            $sRandomKey .= "-SP";
        }
    }

    $sEmailMessage = &get_message($sEmailMessage,
        $sRandomKey,$sPartner,$nUser_ID);

    my %toXdrive =
    (
        To      => "$sFriendsEmail",
        From    => "$sUser_name <$sUser_email>",
        Message => $sEmailMessage,
        Subject => "$sEmailSubject",
    );

    unless (sendmail %toXdrive)
    {
        warn "## Mail error ".$Mail::Sendmail::error;
        if ($Mail::Sendmail::error =~ /451/)
        {
            my $sMessage = $oErr->ReturnMessageGivenCode(1310);
            XDErrorToBrowser("", $sMessage, undef, $oToken);
        }
        else
        {
            my $sMessage = $oErr->ReturnMessageGivenCode(1311);
            XDErrorToBrowser("", $sMessage, undef, $oToken);
        }
        exit(1);
    }
}

sub get_message {
    my ($sEmailMessage, $sRandomKey,$sPartner,$n_UserID) = @_;

    my $oMessage = new XDrive::Template;
    $oMessage->partner($sPartner);
    $oMessage->load('share_a_file__message.thtml');

    $oMessage->tags
    (
        (
            'Message' => $sEmailMessage,
            'RandomKey' => $sRandomKey,
            'nUser_ID' => $n_UserID,
            'sender' => $ENV{'HTTP_HOST'},
        )
    );

    return $oMessage->get;
}

```



```

}

sub display_form {
    my ($sFileName, $bHelp, $sPartner) = @_;
    my $oForm = new XDrive::Template;
    $oForm->partner($sPartner);
    $oForm->load('share_a_file.thtml');

    my $sHelp='';

    if ($bHelp eq 'true')
    {
        my $oHelp = new XDrive::Template;
        $oHelp->partner($sPartner);
        $oHelp->load('share_a_file_help.thtml');
        $sHelp = $oHelp->get;
    }

    $oForm->tags
        (
            ('sFileName' => $sFileName,
             'helptext' => $sHelp
            ));

    print header, $oForm->get;
    exit(0);
}

sub display_thank_you {
    my $sPartner = shift;
    my $oForm = new XDrive::Template;
    $oForm->partner($sPartner);
    $oForm->load('share_a_file__t_y.thtml');
    print header, $oForm->get;
    exit(0);
}

sub verify_database_values {
    my ($nUser_ID, $sFileName, $sFilePath, $sFilePath, $sFileName,
        $sDescription, $oToken, $oErr) = @_;

    if (length($sDescription) > 255) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1320);
        XLErrorToBrowser("", $sMessage, undef, $oToken);
    }

    if (length($sFilePath) > 255) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1321);
        XLErrorToBrowser("", $sMessage, undef, $oToken);
    }

    if (length($sFileName) > 255) {
        my $sMessage = $oErr->ReturnMessageGivenCode(1322);
        XLErrorToBrowser("", $sMessage, undef, $oToken);
    }
}

sub insert_file_into_database {
    my ($nUser_ID, $sFileName, $sFilePath, $sFileDescription,
        $xdDBH, $oToken, $oErr) = @_;

```

```

my @characters = ('a'..'z','A'..'Z','0'..'9');

##seed random number generator
srand(truly_random_value());
my $gmTime = time;
##grab length of time
my $randLen = 32 - length($gmTime);
my $sRandomKey = XDRandomString($randLen,\@characters);
##now we have a Random key
$sRandomKey = $gmTime . $sRandomKey;
## at this point we have a random number
## of length gmTime with the current gmt time appended to it

my $transaction = XDrive::DatabaseO::Transaction->new($xdDBH);
my $status = $transaction->insertDiskItemShare($nUser_ID, $sRandomKey,
$sFilePath, $sFileName, $sFileDescription);

if ($status < 0)
{
    $transaction->rollback();
    my $sMessage = $oErr->ReturnMessageGivenCode(1323);
    XDErrorToBrowser("", $sMessage, undef, $oToken);
    exit(1);
}
else
{
    $transaction->commit();
}

return $sRandomKey;
}

sub get_friends_emails {
my $cgi = shift;
my ($email_list, @email_array);

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail0'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail1'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail2'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail3'));
}

if (length $cgi->param('sFriendsEmail0') > 0)
{
    push(@email_array, $cgi->param('sFriendsEmail4'));
}
}

```

```
$email_list = join(", ", @email_array);  
return $email_list;  
)
```

###signup_account.cgi

```

#!/usr/bin/perl
## -d:DProf
## -d:SmallProf
## Written by Martin Hald <mhald@uci.edu> on Wed Apr 7 1999. This program
## adds new users to the database.
## Modified by Justin White for cookie referee and promo stuff and to make
## mod_perl friendly and to work with changes to the Security module and
## to get rid of the XDrive::CGI module and to create a CGI object.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::Client::Registration;
use XDrive::Error;
use XDrive::Client::Security;
use XDrive::Template;
use XDrive::DatabaseO::Table::UserData;
use XDrive::DatabaseO::Transaction;
use XDrive::DatabaseO::Table::UserQuota;
use XDrive::DatabaseO::Table::Promo;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Table::Reseller;
use XDrive::Template;
use XDrive::DatabaseO::Search;
use XDrive::CGI::Cookie;
use XDrive::Library;
use Mail::Sendmail;
use CGI qw(param redirect header cookie);

BEGIN
{
    push(@INC, "/export/home/www/thirdparty/mint2/perl");
}

use Mint2;

&main;

exit;

sub main {
    my $oCGI = new CGI;
    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $oSTDCookie = XDrive::CGI::Cookie->new('xd_std_info', $oCGI);

    my $file_found;

    ### Use the new XDrive::CGI::Cookie now.
    my $promo_uri = $oCookie->getElement('promo');
    my $ref_seq_cookie = $oCookie->getElement('referee');
    my $referred_from = $oCookie->getElement('referred_from');
    my $claim_ticket = $oCookie->getElement('ct');

    my $ref_seq_param = $oCGI->param('referee');
    my $password = $oCGI->param('password');
    my $password_confirm = $oCGI->param('password_confirm');
    my $birth_year = $oCGI->param('birth_year');

```

```

my $username      = $oCGI->param('username');
my $name_first    = $oCGI->param('name_first');
my $name_last     = $oCGI->param('name_last');
my $email_address = $oCGI->param('email_address');
my $country_seq   = $oCGI->param('country');
my $gender_seq    = $oCGI->param('gender');
my $postal_code   = $oCGI->param('zip2');
my $occupation_seq = $oCGI->param('occupation');
my $referee       = $oCGI->param('referee');
my $marketing     = $oCGI->param('marketing');
my $newsletter    = $oCGI->param('newsletter');
my $media_type_seq = $oCGI->param('media_type');

```

```

## PARAMS TO GATHER IF THIS IS CALLED FROM SKIP
## THE DOWNLOAD

```

```

my $sSTDPartner = $oSTDCookie->getElement('STDPARTNER');
my $sLanguage   = $oSTDCookie->getElement('LANG');
my $sFileURL    = $oSTDCookie->getElement('FILEURL');
my $sFileName   = $oSTDCookie->getElement('FILENAME');
my $sAltURL     = $oSTDCookie->getElement('ALTURL');
my $sCatId      = $oSTDCookie->getElement('CATID');
my $sGid        = $oSTDCookie->getElement('GID');
my $sSid        = $oSTDCookie->getElement('SID');

```

```

## check if database is up

```

```

my $oDBO;
my $oSearch;
if (XDDDBConnectionCheck() && XDNFSCheck())
{
    ## connection good proceed normally
    $oDBO = new XDrive::DatabaseO(undef);
    $oSearch = XDrive::DatabaseO::Search->new($oDBO);
}
else
{
    ## connection bad write data to a temp file and load
    ## upgrading page telling them that they will be
    ## informed once X:drive is up
    $oDBO = undef;
    $oSearch = undef;
    my $tempVar;

```

```

my $tempEmail = $oCGI->param('friends_email1');
my $numFriends = $oCGI->param('numFriends');

```

```

my $addrArray = $tempEmail;
my $nameArray = $oCGI->param('friends_name1');

```

```

## generate list for the javascript array

```

```

for (my $i = 2; $i <= $numFriends; $i++)
{
    $tempVar = $oCGI->param('friends_email' . $i);

    if ($tempVar)
    {
        $addrArray .= "~" . $tempVar;
        $nameArray .= "~" . $oCGI->param('friends_name' . $i);
    }
}

```

```

    reg_while_down (
        $promo_uri,
        $ref_seq_cookie,
        $referred_from,
        $claim_ticket,
        $ref_seq_param,
        $password,
        $birth_year,
        $username,
        $name_first,
        $name_last,
        $email_address,
        $country_seq,
        $gender_seq,
        $postal_code,
        $occupation_seq,
        $referee,
        $marketing,
        $newsletter,
        $media_type_seq,
        $nameArray,
        $addrArray
    );

    ## leave and show upgrading page test me
    print redirect("/upgrading_signup_success.html");
    exit;
}

##my $oDBO      = new XDrive::DatabaseO(undef);
##my $oSearch = XDrive::DatabaseO::Search->new($oDBO);

#####
### If media_type_seq equals 'notset', then set it to NULL.
#####
$media_type_seq = '' if $media_type_seq eq 'notset';

my $partner_code = 'xdrv';
my $partner_seq  = 1;

my $promo_seq;

#####
### Check to see how the referee sequence, if any, was passed in.
### If it was passed in via cookie, then use that. Else, assume
### that it is a form parameter.
#####
my $ref_seq = $ref_seq_cookie ? $ref_seq_cookie : $ref_seq_param;

#####
### If we were passed a promo uri, then let's get the promo seq
### from promo table using Promo.pm to pass to xd_client_register.
#####
if ($promo_uri) {
    my $oPromoInfo = XDrive::DatabaseO::Table::Promo-
>new(undef,$oDBO);
    $oPromoInfo->loadWhere('URI', $promo_uri);
    $promo_seq = $oPromoInfo->fetchColumn('SEQ');

    $oPromoInfo->finish();
}

#####

```

```

### Load the required template HTML files. The content that we load
depends
### on if the new registration went through or if we need to have them
re-fill
### the form.
#####
my $oContent      = new XDrive::Template( {'partner_code' => 'xdrv'} );
my $oLayout       = new XDrive::Template( {'partner_code' => 'xdrv'} );
my $oNavigation   = new XDrive::Template( {'partner_code' => 'xdrv'} );

my $oErr = new XDrive::Error;

$oContent->load('front_signup.thtml');
$oNavigation->load('front_nav.thtml');
$oLayout->load('layout.thtml');

#####
### Perform data validation
#####
if ($password ne $password_confirm) {
    $oErr->AddErrorByErrorCode(709);
}

#####
### Attempt to register the user if no errors have been logged
#####
if (! $oErr->Occurud ) {
    xd_client_register( {'birth_year'      => $birth_year,
                        'partner_seq'      => $partner_seq,
                        'username'        => $username,
                        'password'        => XDEncrypt($password),
                        'name_first'      => $name_first,
                        'name_last'       => $name_last,
                        'email_address'    => $email_address,
                        'country_seq'     => $country_seq,
                        'gender'          => $gender_seq,
                        'postal_code'     => $postal_code,
                        'occupation_seq'   => $occupation_seq,
                        'referee'         => $ref_seq,
                        'marketing'       => $marketing,
                        'newsletter'      => $newsletter,
                        'partner_code'    => $partner_code,
                        'promo_seq'       => $promo_seq,
                        'media_type_seq'  => $media_type_seq},
    $oCGI, $oErr, $oDBO );
}

if ($oErr->MaxIndex() < 0) {
    ## No errors occurred, the user has already been added to the
    ## database through the xd_client_register subroutine so now
    ## send the user an email and then
    ## log the user and go to the user's homepage.

    client_email_send($username,
                      $name_first,
                      $name_last,
                      $email_address,
                      'X\drive Team <team@xdrive.com>',
                      'Welcome to X:drive! - Important Account
Information',
                      $partner_code,
                      $promo_seq);
}

```

```

#####
### If we have a claim ticket, then remove that ticket
### from the batch_user_data table because the user has
### been added and we don't need that data anymore.
#####
if ($claim_ticket) {
    my $oTransaction = XDrive::DatabaseO::Transaction-
>new($oDBO);

    my $rv = $oTransaction->removeClaimTicket($claim_ticket);

    if ($rv == 1) {
        $oTransaction->commit();
    }
    else {
        $oTransaction->rollback();
    }
}

##if we have a referee seq, give the referee additional space
if ($ref_seq >= 1) {

    ## johngaa add to exclude college club and quepasa users
out
    my $oUserData = XDrive::DatabaseO::Table::UserData-
>new(undef,$oDBO);
    $oUserData->loadWhere("SEQ", $ref_seq);
    my $reseller_seq = $oUserData-
>fetchColumn("RESELLER_SEQ");
    if (!(isResellerSeqCC_QUPA($oDBO,$reseller_seq)))
    {

        ## end of johngaa

        my $oUserQuota = XDrive::DatabaseO::Table::UserQuota-
>new(undef, $oDBO);
        $oUserQuota->loadWhere("USER_SEQ", $ref_seq);
        my $additional_quota = $oUserQuota-
>incrementQuota($ref_seq, 5120);

        if ($additional_quota > 0) {

&send_email_referee($ref_seq,$oDBO,$oCookie,$additional_quota,$referred_from)
;

        }

        $oUserQuota->finish();
    }

}

##if the user is from Cybergold, process through Cybergold
if ($promo_uri =~ /cybergold/) {
    my ($code, $res) =
&contact_cybergold($oCGI,$username,$email_address);
}

##if user is coming from the befree promo
##write to file that they've signed up
if ($promo_uri =~ /befree/) {
    &write_befree_log($oCGI);
}

```



```

    }

    if ($sFileURL eq '') {

        client_login($username, $oCGI);

    } else {

        std_login($username,
                  $oCGI,
                  $sSTDPartner,
                  $sLanguage,
                  $sFileURL,
                  $sFileName,
                  $sAltURL,
                  $sCatId,
                  $sGid,
                  $sSid);

    }

    $oSearch->disconnect();

    exit;
}
else {
    ## Reload the signup form, show the errors and pre-fill all
    ## the form elements except the password.

    ##if we are overriding standard registration form
    ##load it here

    if ($promo_uri)
    {
        $file_found = $oContent->load($promo_uri .
'_registration.shtml');
        if (!$file_found)
        {
            $file_found = $oContent-
>load('promo_registration.shtml');
        }
    }

    if ((!$promo_uri) || (!$file_found))
    {
        $oLayout->load("layout.shtml");
        $oNavigation->load("front_nav.shtml");
        $oContent->load("front_signup.shtml");
    }

    my ($select_marketing, $select_newsletter);

    my $checked = "CHECKED";

    if ($marketing eq 'on') {
        $select_marketing = $checked;
    }

    if ($newsletter eq 'on') {
        $select_newsletter = $checked;
    }

    ## IMPORTANT ##
    ## make sure to put all non text fields at the top of

```

```

        ## the tags function or it will gag

        ## Search and replace the following tags
        $oContent->tags( {'country' =>
xd_form_countries($country_seq, $oSearch),
                        'occupation' =>
xd_form_occupation($occupation_seq, $oSearch),
                        'media_type' =>
xd_form_media_type($media_type_seq, $oSearch),
                        'gender' =>
xd_form_gender($gender_seq, $oSearch),
                        'select_marketing' => $select_marketing,
                        'select_newsletter' => $select_newsletter,
                        'errors' => format_errors($oErr),
                        'username' => $username,
                        'name_first' => $name_first,
                        'name_last' => $name_last,,
                        'email_address' => $email_address,
                        'birth_year' => $birth_year,
                        'postal_code' => $postal_code} );

        ##
        ## Added to have tell a friend support in registration
        ##

        my (@addrArray, @nameArray, $tempIndex, $tempName,
$tempEmail, $tempNum);

        ## tell a friend data will be coming in to signup_form
        ## seperated by commas

        @addrArray = split /,/, $oCGI->param('friends_email_array');
        @nameArray = split /,/, $oCGI->param('friends_name_array');

        $tempNum = $oCGI->param('numFriends');

        for (my $tempIndex=1; $tempIndex <= $tempNum; $tempIndex++) {
            $tempName = 'friends_name' . $tempIndex;
            $tempEmail = 'friends_email' . $tempIndex;

            $oContent->tags( {$tempName => $oCGI->param($tempName),
                            $tempEmail => $oCGI->param($tempEmail)}
);

        }

        ## Clear the content of any unused tags.
        $oContent->clear;
    }

    ##if we are loading a non-standard registration, it's only one page
    if (($promo_uri) && ($file_found))
    {
        print $oCGI->header(), $oContent->get;
    }
    else
    {
        ## Print out the HTML and exit
        $oLayout->tags( {'header_graphic' => 'header_registration.gif',
                        'title' => 'Register Now!',
                        'content' => $oContent->get,
                        'navigation' => $oNavigation->get} );
    }

```

```

        print $oCGI->header(), $oLayout->get;
    }

    $oSearch->disconnect();

    return 0;
}

```

```

sub isResellerSeqCC_QUPA
{

```

```

    my $oDBO = shift;
    my $reseller_seq = shift;
    my $dbh = $oDBO->fetchDBH();

    my $sql_stmt = "SELECT code FROM reseller WHERE seq=?";
    my $cmd;
    my @data;

```

```

    $cmd = $dbh->prepare($sql_stmt);
    $cmd->execute(($reseller_seq));
    @data = $cmd->fetchrow_array;
    if ($data[0] eq 'cc' || $data[0] eq 'qupa')
    {
        return 1;
        ##print "should return a true\n"
    }
    return 0;
}

```

```

#####
## reg_while_down:  Grabs all data that is needed to register a user
## routine will add the data to a file in the tmp directory of the name
## reg_while_down.datetime
#####
sub reg_while_down
{

```

```

    my ($promo_uri,$ref_seq_cookie,$referred_from,$claim_ticket,
        $ref_seq_param,$password,$birth_year,$username,$name_first,
        $name_last,$email_address,$country_seq,$gender_seq,$postal_code,
        $occupation_seq,$referee,$marketing,$newsletter,$media_type_seq,
        $tell_a_friend_name,$tell_a_friend_addr) = @_;

```

```

    my $filename = XDGetRegDatFile();
    open OUTFILE, ">>$filename";

```

```

    print OUTFILE "$promo_uri,$ref_seq_cookie,$referred_from,";
    print OUTFILE "$claim_ticket,$ref_seq_param,$password,";
    print OUTFILE "$birth_year,$username,$name_first,";
    print OUTFILE "$name_last,$email_address,$country_seq,";
    print OUTFILE "$gender_seq,$postal_code,$occupation_seq,";
    print OUTFILE "$referee,$marketing,$newsletter,$media_type_seq,";
    print OUTFILE "$tell_a_friend_name,$tell_a_friend_addr\n";
    close OUTFILE;
}

```

```

#####
## format_errors:  Accept an error object and return an ordered list of
## errors in HTML format.
#####

```

```

sub format_errors {
    my $oErr = shift; ## (I) errors

    my $txt;          ## formatted HTML
    my $bPassword;    ## has a password error been found?

    $txt .= "<ol>\n";

    my $nNumErrors = $oErr->MaxIndex();

    for (my $i = 0; $i <= $nNumErrors; $i++) {
        my $error = $oErr->Message();

        if ($error =~ /assword/) {
            $bPassword = 1;
        }

        $txt .= "<li><font color=RED>$error</font>\n";
    }

    if (! $bPassword) {
        $txt .= "<li><font color=RED>Please re-enter your
password</font>\n";
    }

    $txt .= "</ol>\n";

    return $txt;
}

```

```

#####
## client_login: Create the needed token to identify the client and redirect
## them to thier new homepage.
#####

```

```

sub client_login ($$) {
    ## No errors occurred, add the user to the parter/user->real
    ## user mapping and return a success code.

    my $username = shift;
    my $oCGI      = shift;

    my $oDBO      = new XDrive::DatabaseO(undef);
    my $oCookie   = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    #####
    ### Check the x_session_info cookie for promo or referee and
    ### if they exist, delete those hash elements and reset the cookie.
    #####

    my $promo_cookie = $oCookie->getElement('promo');
    my $ref_cookie   = $oCookie->getElement('referee');

    if ($ref_cookie || $promo_cookie) {
        $oCookie->deleteElement('referee') if $ref_cookie;
        $oCookie->deleteElement('promo')   if $promo_cookie;
    }

    print "Set-Cookie: ", $oCookie->asString();
}

my $oError = new XDrive::Error;
my $oToken = xd_login($oCGI, $username, $oError, $oDBO);

```

```

## we need to do all of this to get the reseller code to show the
correct page
my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new(undef,
$oDBO);

$oDiskAccount->loadWhere("USERNAME", $username);

my $oUser = XDrive::DatabaseO::Table::UserData->new(undef,
$oDiskAccount->fetchDBO);

$oUser->loadWherePK($oDiskAccount->fetchColumn("USER_SEQ"));

my $oReseller = XDrive::DatabaseO::Table::Reseller->new(undef,
$oDiskAccount->fetchDBO);

$oReseller->loadWherePK($oUser->fetchColumn("RESELLER_SEQ"));

my $oTemplate = new XDrive::Template;

$oTemplate->partner($oReseller->fetchColumn("CODE"));

## originally this is where the signup_form.cgi goes
##$oTemplate->load('splash.shtml');
$oTemplate->load('tell_a_friend_frame.shtml');

##my $addrArray = $oCGI->param('friends_email_array');
##my $nameArray = $oCGI->param('friends_name_array');
##my $numFriends = $oCGI->param('numFriends');

## generate list for the javascript array
##my @addrList = split /,/, $addrArray;
##my @nameList = split /,/, $nameArray;

##$addrArray = "";
##$nameArray = "";

##my $count = @addrList - 1;

##for (my $i = 0;$i < $count;$i++) {
##    $addrArray .= "\" . $addrList[$i] . "\",";
##    $nameArray .= "\" . $nameList[$i] . "\",";
##}
## this will add the quote without the comma

##$addrArray .= "\" . $addrList[$count] . "\"";
##$nameArray .= "\" . $nameList[$count] . "\"";
## gets the array started
my $tempVar;

my $tempEmail = $oCGI->param('friends_email1');
my $numFriends = $oCGI->param('numFriends');

my $addrArray = "\" . $tempEmail . "\"";
my $nameArray = "\" . $oCGI->param('friends_name1') . "\"";

## generate list for the javascript array
for (my $i = 2;$i <= $numFriends;$i++)
{
    $tempVar = $oCGI->param('friends_email' . $i);

    if ($tempVar)
    {

```

```

    $addrArray .= ", \" . $tempVar . "\"";
    $nameArray .= ", \" . $oCGI->param('friends_name' . $i) . "\"";
}
}

$oTemplate->tags( ('numFriends' => $numFriends,
                  'friends_name_array' => $nameArray,
                  'friends_email_array' => $addrArray) );

print $oCGI->header();

print $oTemplate->get();

$oDiskAccount->finish();
$oUser->finish();
$oReseller->finish();
$oDiskAccount->disconnect();
}

#####
## Login in user who is comming from a Skip The Download
## Registration
#####

sub std_login () {
    my $username = shift;
    my $oCGI = shift;
    my $sSTDPartner = shift;
    my $sLanguage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sCatId = shift;
    my $sGid = shift;
    my $sSid = shift;
    my $oDBO = new XDrive::DatabaseO(undef);

    my $oError = new XDrive::Error;
    my $oToken = xd_login($oCGI, $username, $oError, $oDBO);
    xd_set_session_cookie($oCGI, $sSTDPartner, $sLanguage);

    my $oTemplate = new XDrive::Template
    (
        ('partner_code' => $sSTDPartner,
         'language' => $sLanguage,
         'file' => 'skip_the_download_from_reg.shtml',
         'tags' =>
            {
                'FILE_URL' => $sFileURL,
                'FILE_NAME' => $sFileName,
                'ALTRUL' => $sAltURL,
                'LANG' => $sLanguage,
                'STDPARTNER' => $sSTDPartner,
                'CATID' => $sCatId,
                'GID' => $sGid,
                'SID' => $sSid,
            }
        ));
}

```

```

    $oTemplate->clear();
    print "Content-type: text/html\n\n";
    print $oTemplate->get();
    $oDBO->disconnect();
}

sub contact_cybergold {
    my $oCGI = shift;
    my $msgid = shift;
    my $email = shift;

    my %args = (
        'mint_home'    => $ENV{'MINT_HOME'},
        'msg_mode'     => 'background_mode',

        'usr_email'    => $email,
        'msg_id'       => $msgid,

        'pay_type'     => 'reward',
        'pay_value'    => '1.00',
        'pay_readme'   => 'Thanks for registering with X:drive.',

        'co_name'      => 'X Drive',
        'co_key'       => 'registration',
        'co_account'   => '100500900000396',
        'mint_secret'  => '184FEB9DB81944502A1C91B2879484B6',

        'mint_url_pay' => 'http://www1.cybergold.com/payserver?pay_server',
        'msg_version'  => '2.2'
    );

    my($code, %res) = mint_invoke(\%args);

    ##this is temp code to print out stuff for cybergold
    ##my @keys = keys %res;
    ##my @values = values %res;
    ##while (@keys)
    ##{
    ##    die pop(@keys), '=', pop(@values), "\n";
    ##}

    return $code;
}

sub write_befree_log {
    my $oCGI = shift;

    my $source_id = $oCGI->cookie('sourceid');

    ##get the time
    ##needed to figure out name of file to write to
    my ($nSec, $nMin, $nHour, $nDay, $nMonth, $nYear, $sDay) =
        (localtime(time))[0,1,2,3,4,5,6];

    if ($nYear > 99) {
        $nYear = substr($nYear,1,2);
    }

    ## Numeric month is 0-11, so add one
    $nMonth++;

    ## Handle Y2K issue

```

```

        if ( $nYear >= 80 ) {
            $nYear += 1900;
        }
        else {
            $nYear += 2000;
        }

        my $dToday = sprintf("%s%02d%02d", $nYear, $nMonth, $nDay);
        my $dTodayFull = sprintf("%02d%02d%s%02d:%02d:%02d", $nMonth, $nDay, $nYear, $nHour, $nMin, $nSec);

        my $text =
"14524098\tS\t$dTodayFull\t$source_id\tl\tl\tl\t0.00\tUSD\tregistration\n";

        warn "#BF", $text, "\n";
        ##open(FILE, ">>xdrive_orders_$dToday.txt");
        ##print FILE $text;
        ##close(FILE);
    }

sub send_email_referee {
    my $user_seq = shift;
    my $oDBO = shift;
    my $oCookie = shift;
    my $additional_quota = shift;
    my $referred_from = shift;

    my $language = $oCookie->getElement('language');
    my $partner = $oCookie->getElement('partner');

    if ($language eq 'spanish') {
        my $text = 'un amigo que usted refirió';
        if ($referred_from eq '2') {
            $text = 'un usted compartió un fishero con';
        }
    }
    else {
        my $text = 'referred';
        if ($referred_from eq '2') {
            $text = 'shared a file with';
        }
    }

    my $text = 'referred';
    if ($referred_from eq '2') {
        $text = 'shared a file with';
    }

    ##comes in as k, change to megabytes
    my $mbs = $additional_quota/1024;

    my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef,
$oDBO);
    $oUserData->loadWhere("SEQ", $user_seq);
    my $email_address = $oUserData->fetchColumn("EMAIL_ADDRESS");
    my $name_first = $oUserData->fetchColumn("NAME_FIRST");
    my $name_last = $oUserData->fetchColumn("NAME_LAST");

    my $oTemplate = new XDrive::Template( {'language' => $language,
'partner_code' => $partner} );

    $oTemplate->load('received_5MB_tellafriend.shtml');

```



```
$oTemplate->tags( {'mbs' => $mbs,
                  'text' => $text} );
$oTemplate->clear();

my $message = $oTemplate->get;

my %toXdrive =
(
  To      => "$name_first $name_last <$email_address>",
  Bcc     => '',
  From    => "support\@xdrive.com",
  Message => $message,
  Subject => "Congratulations!"
);

sendmail(%toXdrive);

$oUserData->finish();
}
```

###signup_form.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999. Updated
## Fri Apr 5, 1996 to use new templates. Updated Wed Apr 21 1999 to use
## new library code.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});
```

```
use CGI;
use CGI::Carp 'fatalsToBrowser';
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Registration;
use XDrive::Template;
use XDrive::DatabaseO::Search;
use XDrive::Library;
```

```
use constant XD_REGISTRATION_DEFAULT_COUNTRY => 223;
```

```
exit &main;
```

```
sub main {
    my $oContent      = new XDrive::Template;
    my $oNavigation    = new XDrive::Template;
    my $oLayout        = new XDrive::Template;
    my $oCGI           = new CGI;

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    my $oSearch;

    my $sReferee      = $oCGI->param('referee');
    my $sClaimTicket  = $oCookie->getElement('ct');

    ## Defaults
    my $sUsername      = undef;
    my $sNameFirst     = undef;
    my $sNameLast      = undef;
    my $nYOB           = undef;
    my $nPromoSeq      = undef;
    my $nGender         = 3;
    my $sEmailAddress  = undef;

    my ($country_seq, $occupation_seq, $postal_code, $ct_promo_seq);

    my %pullDownHash;
    if (XDDBCConnectionCheck() && XDNFSCheck())
    {
        $oSearch = XDrive::DatabaseO::Search->new(undef);
    }
    else
    {
        $sClaimTicket = undef;
        $oSearch = undef;
        %pullDownHash = generate_db_array();
    }
    if ($sClaimTicket) {
        my $rhData = getUserData($oSearch, $sClaimTicket);
```

```

    if ($rhData) {
        my $oNewCgi = CGI->new($rhData);

        $sUsername      = $oNewCgi->param('username');
        $sNameFirst      = $oNewCgi->param('name_first');
        $sNameLast       = $oNewCgi->param('name_last');
        $sEmailAddress   = $oNewCgi->param('email_address');
        $nYOB            = $oNewCgi->param('birth_year');
        $nGender         = $oNewCgi->param('gender');
        $soccupation_seq = $oNewCgi->param('occupation_seq');
        $sCountry_seq    = $oNewCgi->param('country_seq');
        $sPostal_code    = $oNewCgi->param('postal_code');
    }

    if ($sReferee ne "") {
        # my $oCookie = XDrive::CGI::Cookie->new('x_session_info',
SoCGI);
        my $sReferred_from = $oCGI->param('type');
        $oCookie->setElement({'partner_code'=>'xdrv'});
        $oCookie->setElement({'language'=>'english'});
        $oCookie->setElement({'referee' => $sReferee});
        $oCookie->setElement({'referred_from' => $sReferred_from});
        print "Set-Cookie: ".$oCookie->asString();
    }

    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');
    $oLayout->partner('xdrv');

    ## I'm assuming there will be one page and not a series of frames.
    ## this can be changed if need be
    # my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    # my $promo = $oCookie->getElement('promo');

    my $promo = $oCookie->getElement('promo');

    my $file_found;

    ##if we have a promo, try to get a special registration page
    if ($promo) {
        ##attempt to open a special registration page
        $file_found = $oLayout->load($promo . '_registration.shtml');
        if (!$file_found) {
            ##if we cannot, open the general promo reg page
            $file_found = $oLayout->load('promo_registration.shtml');
        }
    }

    ##is we don't have a promo then use the standard registration
    if ( (! $promo) || (! $file_found) ) {
        ## Load the required template HTML files.
        $oNavigation->load("front_nav.shtml");
        $oContent->load("front_signup.shtml");
        $oLayout->load("layout.shtml");

        $oContent->tags
        (
            'username'      => $sUsername,
            'name_first'    => $sNameFirst,
            'name_last'     => $sNameLast,
            'email_address' => $sEmailAddress,

```

```

        'country' =>
xd_form_countries_db_check(XD_REGISTRATION_DEFAULT_COUNTRY,
$oSearCh,\%pullDownHash),
        'occupation' => xd_form_occupation_db_check(undef,
$oSearCh,\%pullDownHash),
        'media_type' => xd_form_media_type_db_check(undef,
$oSearCh,\%pullDownHash),
        'gender' => xd_form_gender_db_check(undef,
$oSearCh,\%pullDownHash),
        'select_marketing' => 'CHECKED',
        'select_newsletter' => 'CHECKED',
        'referee' => $sReferee,
    });

## Print out the HTML and exit
$oSLayout->tags
    (
        'header_graphic' => 'header_registration.gif',
        'title' => 'Register Now!',
        'content' => $oSContent->get,
        'navigation' => $oSNavigation->get
    );
}
elseif ($sClaimTicket) {
    $oSLayout->tags
        (
            'country' => xd_form_countries($sCountry_seq,
$oSearCh),
            'occupation' => xd_form_occupation($sOccupation_seq,
$oSearCh),
            'media_type' => xd_form_media_type(undef, $oSearCh),
            'gender' => xd_form_gender($nGender, $oSearCh),
            'select_marketing' => 'CHECKED',
            'select_newsletter' => 'CHECKED',
            'username' => $sUsername,
            'name_first' => $sNameFirst,
            'name_last' => $sNameLast,
            'email_address' => $sEmailAddress,
            'birth_year' => $nYOB,
            'referee' => $sReferee,
            'postal_code' => $postal_code
        );
}
else {
    $oSLayout->tags
        (
            'country' =>
xd_form_countries_db_check(XD_REGISTRATION_DEFAULT_COUNTRY,
$oSearCh,\%pullDownHash),
            'occupation' => xd_form_occupation_db_check(undef,
$oSearCh,\%pullDownHash),
            'media_type' => xd_form_media_type_db_check(undef,
$oSearCh,\%pullDownHash),
            'gender' => xd_form_gender_db_check(undef,
$oSearCh,\%pullDownHash),
            'select_marketing' => 'CHECKED',
            'select_newsletter' => 'CHECKED',
            'referee' => $oCGI->param('referee'),
        );
}

$oSLayout->clear;

```

```

        print $oCGI->header, $oLayout->get;
        if (defined $oSearch)
        {
            $oSearch->disconnect();
        }

        return 0;
    }

## johngaa add to check of db is up or down
sub generate_db_array
{
    ## create a hash
    my %tempHash;
    my $i = 1;
    my $key;
    my @tempVal;
    open FH, "<down_data.dat";

    while(<FH>)
    {
        chomp $_;
        if ($_ =~ /^#(\w+)/g)
        {
            my @newArray;
            $i = 1;
            $key = $_;
            $tempHash{$key} = [ @newArray ];
        }
        else
        {
            @tempVal = split(/\~/, $_);

            $tempHash{$key}->[$i - 1][0] = $tempVal[0];
            $tempHash{$key}->[$i - 1][1] = $tempVal[1];
            $i++;
        }
    }

    close FH;
    return %tempHash;
}

sub xd_form_countries_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_countries(XD_REGISTRATION_DEFAULT_COUNTRY,
        $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'country'};
        $returnVal = options_list(XD_REGISTRATION_DEFAULT_COUNTRY, @$templ);
    }
}

```

```
    return $returnVal;
}

sub xd_form_occupation_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_occupation(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'occupation'};
        $returnVal = options_list(undef, @$templ);
    }

    return $returnVal;
}

sub xd_form_media_type_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_media_type(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'media_type'};
        $returnVal = options_list(undef, @$templ);
    }

    return $returnVal;
}

sub xd_form_gender_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_gender(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'gender'};
        $returnVal = options_list(undef, @$templ);
    }
}
```

```

    return $returnVal;
}

```

```

## end of johngaa add

```

```

sub getPromoURI ($$) {
    my $oSearch = shift;
    my @promo_seq = (shift);

    my $oDBH = $oSearch->fetchDBO->fetchDBH();

    my $st = "SELECT uri FROM xdrive.promo WHERE seq = ?";

    my $data = $oDBH->selectcol_arrayref($st, undef, @promo_seq);

    return $data->[0];
}

```

```

sub getUserData {
    my $oSearch = shift;
    my $sTicket = shift;

    my $oDBH = $oSearch->fetchDBO->fetchDBH();
    my $sQuery = "SELECT DATA FROM BATCH_USER_DATA WHERE CODE = ?";
    my $oCursor = $oDBH->prepare($sQuery);
    $oCursor->bind_param(1, $sTicket);
    $oCursor->execute;

    my $rh;
    my $sData = $oCursor->fetchrow_array();
    # my ($sData) = $oCursor->fetchrow_array();
    # eval $sData;
    # return $rh;
    return $sData;
}

```

###signup_success.cgi

```
#!/usr/bin/perl
## This CGI allows us to pass the sst and sid on to the inner frame
##
## Modified by Justin White on 10/14/99 by manually printing the
## header to the browser and getting rid of the XDrive::CGI import.
## Created new cgi, database, and error objects to pass to xd_security_check.
## Also added the exit in the sub call.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI::Carp qw(fatalsToBrowser);
use CGI ();
use Token;
use XDrive::Client::Security;
use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::Library;
use XDrive::CGI::Cookie;

&main();

exit;

sub main
{
    my $oCGI      = new CGI;
    my $oDBO      = new XDrive::DatabaseO;
    my $oErr      = new XDrive::Error;
    my $oCookie   = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    #####
    ## Attempt to authenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If the authentication fails or there is an error during the
    ## authentication phase then redirect to the error CGI
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    #####
    ## Otherwise we have a valid session
    #####

    my $sUsername = $oToken->data('user');

    ### Edited by Justin so that the partner_code is looked for in
    ### the cookie instead of the token table.
    # my $sPartner = $oToken->data('partner_code');
    my $sPartner = $oCookie->getElement('partner');
```



```
    if (! defined $sPartner)
    {
        $sPartner = "xdrv";
        $oCookie->setElement({'partner'=>$sPartner});
        print "Set-Cookie: ", $oCookie->asString();
    }

    my $oTemplate = new XDrive::Template( {'partner_code' => $sPartner}
);

    $oTemplate->load('signup_success.shtml');
    $oTemplate->tags( {'username' => $sUsername} );

    print "content-type: text/html\n\n";

    print $oTemplate->get();

    $oDBO->disconnect();

    return 0;
}
```

###signup_toc.cgi

```

#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999. Updated
## Fri Apr 5, 1996 to use new templates.
##
## Modified by Justin White on 10/11/1999 so that it sets a cookie.
##
## Modified by Martin Hald on 11/15/1999 so that it now accepts
## - partner
## - language
## - agreeuri
## - disagreeuri

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::Template;
use XDrive::CGI::Cookie;

&main();

exit;

sub main {
    my $cookie;
    my $sPartnerCode;

    my $oCGI = new CGI;
    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $sReferee = $oCGI->param('referee');
    my $sPartner = $oCGI->param('partner');
    my $sLanguage = $oCGI->param('language');
    my $sReferred_from = $oCGI->param('type');

    $oCookie->setElement({'partner_code'=>$sPartner});
    $oCookie->setElement({'language'=>$sLanguage});

    if ($sReferee ne "") {
        $oCookie->setElement({'referee' => $sReferee});
        $oCookie->setElement({'referred_from' => $sReferred_from});
        print "Set-Cookie: ".$oCookie->asString();
    }

    if (! defined $sPartner) {
        $sPartner = 'xdrv';
    }

    ## Load the terms and conditions
    my $hDefaults = {'partner_code'=>$sPartner, 'cookie'=>$oCookie};
    my $oContent = new XDrive::Template($hDefaults);
    my $oLayout = new XDrive::Template($hDefaults);

    $oContent->load('presignup.shtml');

    if ($sPartner eq 'xdrv') {
        my $oNavigation = new XDrive::Template($hDefaults);
        my $oHeader = new XDrive::Template($hDefaults);
        my $oFooter = new XDrive::Template($hDefaults);
    }
}

```

```
$oLayout->load('layout.shtml');
$oNavigation->load('front_nav.shtml');
$oHeader->load('presignup_header.shtml');
$oFooter->load('presignup_footer.shtml');
$oContent->tags({'header' => $oHeader->get,
               'footer'  => $oFooter->get, });
$oLayout->tags({'navigation' => $oNavigation->get,
               'header_graphic' => 'header_registration.gif',});
} else {
    $oLayout->load('tac_wrapper.shtml');
}

my $sAgreeURI    = $oCGI->param('agreeuri');
my $sDisagreeURI = $oCGI->param('disagreeuri');

$oLayout->tags({'title'      => 'Terms and Conditions',
               'content'    => $oContent->get,
               'agreeuri'   => $sAgreeURI,
               'disagreeuri' => $sDisagreeURI,});
$oLayout->clear;

print $oCGI->header();
print $oLayout->get;

return 0;
}
```

###skip_the_download.cgi

```

#!/usr/bin/perl

use strict;
use lib $ENV{PERL_XDRIVE_LIB};

use CGI qw(param redirect header cookie);
use CGI::Cookie;

use LWP::UserAgent;

use CGI::Carp qw(fatalsToBrowser);
use XDrive::Client::Security;
use XDrive::Client::Actions;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Search;
use XDrive::DatabaseO::Transaction;
use XDrive::Template;
use XDrive::CGI qw(:MAIN);
use XDrive::CGI::Cookie;
use XDrive::DatabaseO;
use XDrive::Error;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;
use Token;

my $oDBO = new XDrive::DatabaseO;
main($oDBO);

$oDBO->disconnect;
exit;

#####
## NOTE: Remove the quota check from here. will be handled in java.
#####

sub main
{
    my $oDBO = shift;
    my $oCGI = CGI->new();
    my $oErr = new XDrive::Error;
    my $oCookie = XDrive::CGI::Cookie->new('xd_std_info', $oCGI);

    ## params for file url and file name
    my $sFileURL = $oCGI->param('FILEURL');
    my $sFileName = $oCGI->param('FILENAME');
    my $sAltURL = $oCGI->param('ALTURL');
    my $sSid = $oCGI->param('SID');
    my $sGid = $oCGI->param('GID');
    my $sCatId = $oCGI->param('CATID');
    my $sPartnerCode = $oCGI->param('STDPARTNER');
    my $sLanguageCode = $oCGI->param('LANG');
    my $sUsername = $oCGI->param('user');
    my $sPassword = $oCGI->param('pass');
    my $sError = $oCGI->param('error');
    my $sCookie = $oCGI->cookie('SST');

```

```

my $sessionCookie;
my $sPromo = '';
my $sPartnerParams = "";
my $sCNetString = "";

## IF THE SPECIAL C|NET VARIABLES ARE DECLARED
## THEN GENERATE THE C|NET STRING
## THIS URL IS CALLED FOR ANY FILE DOWNLOADED
## FROM C|NET SO THAT THEY CAN CREDIT THE FILE
## BEING DOWNLOADED
if (
    ($sSid != '') &&
    ($sGid != '') &&
    ($sCatId != '')
) {

    $sAltURL = "http://beta.cnet.com/downloads/0-" . $sCatId . "-107-"
. $sSid . ".html?tag=ex.dl.xdrive";

    ## IF YOU ARE ON THE TEST SERVERS,
    ## THEN USE C|NET'S TEST URL
    if (
        ($ENV{'HTTP_HOST'} eq 'martini.xdrive.com') ||
        ($ENV{'HTTP_HOST'} eq 'antifreeze.xdrive.com')
    ){

        $sCNetString = "http://abv-sjc2-
export2.cnet.com/downloads/0,10152,0-" .
            $sCatId .
            "-110-" .
            $sSid .
            ",00.html?gid=" .
            $sGid .
            "&tag=ex.dl.xdrivepop.dlcgi." .
            $sSid;

        ## ELSE, USE THEIR REAL URL
    } else {

        $sCNetString = "http://abv-sjc1-
export2.cnet.com/downloads/0,10152,0-" .
            $sCatId .
            "-110-" .
            $sSid .
            ",00.html?gid=" .
            $sGid .
            "&tag=ex.dl.xdrivepop.dlcgi." .
            $sSid;

    }

}

}

$sPartnerParams =
"STDPARTNER=$sPartnerCode&LANG=$sLanguageCode&ALTURL=$sAltURL";

$soCookie->setElement(
    {
        'FILEURL'    => $sFileURL,
        'FILENAME'   => $sFileName,
    }
)

```

```

        'ALTURL'      => $sAltURL,
        'STDPARTNER' => $sPartnerCode,
        'LANG'        => $sLanguageCode,
        'CATID'        => $sCatId,
        'SID'          => $sSid,
        'GID'          => $sGid,
    ));

    print "Set-Cookie: ". $oCookie->asString();

    my $n = 0;
    my $rv;

    ## Create the database object
    my $oSearch = XDrive::DatabaseO::Search->new($oDBO);

    ##The token for the user session
    my $oToken;

    ## If u/p
    if (defined $sUsername && defined $sPassword)
    {
        ## Auth or fail
        if (xd_auth_password($sUsername, $sPassword, $oDBO))
        {
            $oToken = xd_login($oCGI,$sUsername,$oErr);
            $sessionCookie = xd_set_session_cookie($oCGI,
            $sPartnerCode, $sLanguageCode, $sPromo);
        }
        else
        {
            ## Login failed
            my $r = getHTMLContent
            (
                'skip_the_download_login_failed.shtml',
                $sFileURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
            );

            print "Content-type: text/html\n\n";
            print $r;
            return 1;
        }
    }

    ## error or cookie not defined
    elsif ( (length($sError) > 0) || (length($sCookie) == 0) )
    {
        ## show the login page
        my $r = getHTMLContent('skip_the_download_login.shtml',
                                $sFileURL,
                                $sFileName,
                                $sAltURL,
                                $sPartnerCode,
                                $sLanguageCode
                            );

        print "Content-type: text/html\n\n";
        print $r;
    }
}

```

```

        return 1;
    }
    else
    {
        ## cookie defined so authenticate it
        {
            $oToken = xd_security_check($oDBO,$oCGI,$oErr);
            $sessionCookie = xd_set_session_cookie($oCGI, $sPartnerCode,
            $sLanguageCode, $sPromo);

            if ($oErr->Occurud)
            {
                print $oCGI->redirect("/cgi-
bin/skip_the_download.cgi?&error=expired&$sPartnerParams");
                return 1;
            }
        }

        if (!$sFileURL) {
            my $thtml = ($sAltURL != '')?
'skip_the_download_no_alt_error.thtml'
                        : 'skip_the_download_error.thtml';

            my $sMessage = $oErr->ReturnMessageGivenCode(1220);

            &ThtmlErrorOut($thtml,
                $sMessage,
                $sFileURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
            );
        }

        ## create the Actions object and download the file
        my $oAction = new XDrive::Client::Actions($oToken,$oCGI);

        ## set the filename and file url
        $oAction->STDFilename($sFileName);
        $oAction->STDURL($sFileURL);

        ## see if file exists. if yes, give em message
        my $bFileExists = $oAction->STDFileExists();

        if ($bFileExists)
        {
            $oDBO->disconnect();
            my $sMessage = $oErr->ReturnMessageGivenCode(1242);

            ErrorOut($sMessage,$sFileURL,$sFileName,$sAltURL,$sPartnerCode,$sLangua
geCode);
        }

        ## Check that the file is not already being downloaded
        if ($oSearch->XDSTDBeingDownloaded($oToken->user,$sFileURL))
        {
            $oDBO->disconnect();
            my $sMessage = $oErr->ReturnMessageGivenCode(1243);

```

```

    ErrorOut ($sMessage,$sFileURL,$sFileName,$sAltURL,$sPartnerCode,$sLanguageCode);
}

## Spool the action to download the file
my $oTransaction = new XDrive::DatabaseO::Transaction($oDBO);
my $nSeq = $oTransaction->insertSkipTheDownload
(
    $oToken->user,
    $sFileName,
    $sFileURL,
    0,
    undef
);
$oTransaction->commit;

## Insert failed return an error
if ($nSeq < 0)
{
    $oDBO->disconnect();
    my $sMessage = $oErr->ReturnMessageGivenCode(1244);
    ErrorOut ($sMessage,$sFileURL,$sFileName,$sAltURL,$sPartnerCode,$sLanguageCode);
}

## IF THE INSERT DIDN'T FAIL,
## AND THE SPECIAL C|NET URL ISN'T NULL
## THEN CREDIT C|NET
elsif ($sCNetString ne '')
{
    my $oUA = new LWP::UserAgent;
    $oUA->agent("XDriveSTD/0.1 " . $oUA->agent);

    # Create a request
    my $oRequest = new HTTP::Request GET => $sCNetString;

    # Pass request to the user agent and get a response back
    my $oResult = $oUA->request($oRequest);
}

print redirect("/cgi-bin/skip_the_download_status.cgi?seq=$nSeq&$sPartnerParams");
}

sub ErrorOut ()
{
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

```



```

my $html = &getHTMLContent('skip_the_download_no_alt_error.shtml',
                           $sFileURL,
                           $sFileName,
                           $sAltURL,
                           $sPartnerCode,
                           $sLanguageCode,
                           $sMessage,
                           );

print "Content-type: text/html\n\n";
print $html;
exit(0);
}

```

```

sub ThtmlErrorOut ()
{
    my $thtml = shift;
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    my $html = &getHTMLContent($thtml,
                               $sFileURL,
                               $sFileName,
                               $sAltURL,
                               $sPartnerCode,
                               $sLanguageCode,
                               $sMessage,
                               );

    print "Content-type: text/html\n\n";
    print $html;
    exit(0);
}

```

```

sub getHTMLContent
{
    my $thtmlfile = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sMessage = shift;

    my $template = new XDrive::Template
    (
        (
            'partner_code' => $sPartnerCode,
            'language' => $sLanguageCode,
            'file' => $thtmlfile,
            'tags' =>
            (
                'FILE_URL' => $sFileURL,
                'FILE_NAME' => $sFileName,
                'ALTURL' => $sAltURL,
                'LANG' => $sLanguageCode,
            )
        )
    );
}

```

```
'STDPARTNER' => $sPartnerCode,
'message' => $sMessage,
}
));

$template->clear();

return $template->get;
}

## Create a string which makes the previously created
## cookie expire.

sub empty_cookie
{
    my $oSelf = shift;
    my $cookie = new CGI::Cookie
    (
        -name      => 'sst',
        -value     => '',
        -expires   => '-1M'
    );
    print header(-cookie=>[$cookie]);
}
```

###skip_the_download_status.cgi

```

#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header redirect);
use XDrive::CGI;
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::Database0;
use XDrive::Database0::Table::SkipDownload;
use XDrive::Template;
use XDrive::Error;
use XDrive::Library;
use Token;

use strict;

use constant TEMP_DIR => XDSTDTempDirectory();

&main;
exit(0);

sub main
{
    ## get parameters
    my $nFileSize;
    my $sTempFile;
    my $sFileName;
    my $sError;
    my $nStatus;
    my $bDone;
    my $percent = 0;
    my $nDownloadedSize = 0;
    my $sURL;
    my $nNow;

    my $oCGI = new CGI();
    my $nSeq = $oCGI->param('seq');
    my $nStart = $oCGI->param('start');
    my $sPartnerCode = $oCGI->param('STDPARTNER');
    my $sLanguageCode = $oCGI->param('LANG');
    my $sAltURL = $oCGI->param('ALTURL');
    my $previous_percent = $oCGI->param('pp');

    ## SET THE CONNECTION_COUNT = 0 IF IT ISN'T PASSED IN
    my $connection_count = ($oCGI->param('cc')) ? $oCGI->param('cc') : 0;

    my $oErr = new XDrive::Error;

    ## get the token and the action object
    my $oDBO = new XDrive::Database0;
    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);
    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);

    my $sPartnerParams =
"STDPARTNER=$sPartnerCode&LANG=$sLanguageCode&ALTURL=$sAltURL";

    if ($oErr->Occurud)

```

```

    {
        print redirect("/cgi-bin/skip_the_download.cgi?${sPartnerParams}");
        return;
    }

    ## if the sequence number was passed then get information from the
    database.
    if (defined $nSeq)
    {
        ## load the information from the database
        my $oSkip = XDrive::DatabaseO::Table::SkipDownload->new(undef, $oDBO);
        $oSkip->loadWhere('SEQ',$nSeq);
        $nFileSize = $oSkip->fetchColumn('FILE_SIZE_BYTES');
        $sTempFile = $oSkip->fetchColumn('FILENAME_FOR_TEMP_FILE');
        $sFileName = $oSkip->fetchColumn('FILE_NAME');
        $nStatus = $oSkip->fetchColumn('IS_ACTIVE');
        $sError = $oSkip->fetchColumn('ERROR_CODE');
        $sURL = $oSkip->fetchColumn('FILE_URL');
        $bDone = $oSkip->fetchColumn('IS_DONE');
    }

    ## XDRIVE.SKIP_THE_DOWNLOAD.IS_ACTIVE legend
    ## 0 - still in queue
    ## 1 - being downloaded
    ## 2 - on hold

    ## IF CONNECTION_COUNT > 9, THEN GO TO THE FILE NOT FOUND (1220) ERROR
    ## DISPLAY, BUT KEEP TRYING TO DOWNLOAD THE FILE
    if ($connection_count > 9) {
        $sError=1220;
    }

    ## IF AN ERROR OCCURRED THEN DISPLAY IT
    ## AND THEN EXIT(0);
    if (defined $sError)
    {
        if ($sError == 1240)
        {
            &DisplayQuotaError('',
                                $sURL,
                                $sFileName,
                                $sAltURL,
                                $sPartnerCode,
                                $sLanguageCode
                                );
        }
        else
        {
            my $oErr = new XDrive::Error;
            $oErr->AddErrorByErrorCode($sError);
            &DisplayError($oErr->Message(),
                            $sURL,
                            $sFileName,
                            $sAltURL,
                            $sPartnerCode,
                            $sLanguageCode
                            );
        }
    }

    ## IF THERE IS NO ERROR, THEN GATHER STATUS
    ## AND DISPLAY TO THE USER

```

```

else
{

## Get file size, later change to get from a tmp file
my $sPath = TEMP_DIR."/ $sTempFile";

## IF STATUS IS LISTED AS DONE IN THE DB,
## THEN SHOW THE DONE PAGE
if ($bDone == 1)
{
    &DisplayDone('',
                $sURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
            );
}

## ELSE FILE IS NOT DONE,
## GATHER MORE DATA AND DISPLAY TO USER
else
{

## IF STATUS IS NOT ACTIVE, OR THE FILE DOESN'T EXIST
## THEN DISPLAY THE CONTACTING SERVER PAGE
## REMOVED: || ! -e $sPath
## FROM CHECK
if ( ($nStatus == 0 || -e $sPath)
    && (!($previous_percent >= 0))
    )
{
    &DisplayContactServer($nSeq, $sURL, $sFileName, $sAltURL, $sPartnerCode, $sL
anguagCode, $sPartnerParams, $connection_count);
}

## ELSE, GATHER STATUS DATA
## AND DISPLAY TO USER
else
{

## Set the start time in seconds since the epoch if not passed
## as parameter
if (! defined $nStart || $nStart !~ /\d+$/)
{
    $nStart = time();
}

## IF NO FILE SIZE HAS BEEN SET IN THE DB
## DISPLAY ZERO PERCENTAGES TO THE USER
if (! defined $nFileSize || $nFileSize == 0)
{
    $nFileSize = '0';
    $percent = '0';
    &DisplayStatus($nSeq, $percent, $sFileName, $nFileSize, '',
                $nStart, '', '',
                $sAltURL, $sPartnerCode, $sLanguageCode, $sPartnerParams);
}
}
}

```

```

    }

    ## ELSE
    ## * THERE WAS NO ERROR
    ## * THE FILE WAS NOT DONE
    ## * THE FILE EXISTS IN THE TEMPORARY DIRECTORY
    ## * THE DB HAS AN EXPECTED FILE SIZE
    ## SO READ THE FILE, CALCULATE DATA, AND DISPLAY TO USER
    else
    {

        ## These checks are performed before inserting the skip
information      ## into the database, but we will do it again here to be
safe.

#       my $sError = $oErr->ReturnMessageGivenCode(141);
#       XDErrorToBrowser("", $sError, undef, $oToken);
        ##die "Cannot check $sPath" if $sPath =~ /\.\./;
        ##die "Cannot check $sPath" if $sPath =~ /\//;

        ## Get the size of the download object
        my @file_info = stat($sPath);

        ## Conver the downloaded file size into KB
        if ($file_info[7] > 0)
        {
            $nDownloadedSize = $file_info[7];

            if ($nFileSize > 0)
            {
                $percent = 100 * $nDownloadedSize/$nFileSize;
            }
            if ($percent < 0)
            {
                $percent = 0;
            }
            $percent = sprintf("%.2f", $percent);
        }

        ## IF THE FILE IS GONE NOW, OR SOMEOTHER CONDITION, THE USER
        ## WILL NEVER SEE THE %DONE DROP
        ## USE WHICH EVER IS LARGER, THE PRECENT THAT WE JUST
DISPLAYED      ## OF THE ONE THAT WE JUST READ FROM THE FILE SYSTEM
        $percent = ($previous_percent > $percent) ? $previous_percent
: $percent;

        ## We have already transfered some of the file, so we can now
        ## estimate the download time.
        $nNow = time();

        my $sInfo;
        my $nElapsedSec = $nNow - $nStart;
        my $nTransPerSec = 0;

        if ($nElapsedSec)
        {
            $nTransPerSec = $file_info[7]/$nElapsedSec;
        }

        if ($nTransPerSec > 0)

```

```

{
    my $partial = $percent/100;
    my ($nSecsRemain, $nMin, $nSecs, $nTransPerSecMB);

    if ($partial == 0) {
        $sInfo = '';
    } else {
        $nSecsRemain = ($nElapsedSec/$partial)-$nElapsedSec;
        $nMin = int($nSecsRemain/60);
        $nSecs = $nSecsRemain % 60;
        $nTransPerSecMB = $nTransPerSec/1024;
    }

    $sInfo = sprintf(", %d:%02d remaining (%.2f
KB/sec)", $nMin, $nSecs
                                , $nTransPerSecMB);
}

my $nTrans;

my $k = "KB";
my $nDiv = 1024;
my $nTempSize = $file_info[7] || 0;

if ($nFileSize > 1024*1024)
{
    $k = "MB";
    $nDiv = 1024*1024;
}

if ($nFileSize < 0)
{
    $nFileSize = 0;
}

$nFileSize = sprintf("%.2f", $nFileSize/$nDiv);
$nTrans = sprintf("%.2f", $nTempSize/$nDiv);

&DisplayStatus($nSeq, $percent, $sFileName, $nFileSize, '',
                $nStart, $sInfo, $k,
                $sAltURL, $sPartnerCode, $sLanguageCode, $sPartnerParams);

    ## END OF READING DATA FROM SYSTEM AND
    ## DISPLAYING TO USER
}

    ## END OF NO EXPECTED SIZE IN DB
    ## SHOW USER ZERO PERCENTAGES
}

    ## END OF FILE MUST BE DONE
    ## SO SHOW A DONE
}

    ## END OF NO ERROR
}

    $oDBO->disconnect;
}

```

```

sub DisplayContactServer
{
    my
    ($nSeq,$sURL,$sFileName,$sAltURL,$sPartnerCode,$sLanguageCode,$sPartnerParams
    ,$connection_count) = @_;

    my ($sHostname) = $sURL =~ /\:\/\/([^\/]+)\//;
    $connection_count++;

    ## load the status page
    my $template = new XDrive::Template
    (
        {
            'partner_code' => $sPartnerCode,
            'language' => $sLanguageCode,
            'file' => 'skip_the_download_contacting.shtml',
            'tags' =>
            {
                'hostname' => $sHostname,
                'continue_to' => "/cgi-
bin/skip_the_download_status.cgi?seq=$nSeq&cc=$connection_count&$sPartnerPara
ms",
                'fileName' => $sFileName,
                'altURL' => $sAltURL,
            }
        }
    );
    print "Content-type: text/html\n\n";
    print $template->get;
}

```

```

sub DisplayStatus
{
    my $nSeq = shift;
    my $percent = shift;
    my $filename = shift;
    my $filesize = shift;
    my $stransferred = shift;
    my $start = shift;
    my $info = shift;
    my $k = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sPartnerParams = shift;

    my $percent_disp;

    if ($filesize <= 0)
    {
        $filesize = 'Unknown';
        $k = ' ';
        $percent_disp = 'Unknown';
        $percent = 0;
    }
    else
    {
        $percent_disp = "$percent%";
    }

    ## load the status page
    my $template = new XDrive::Template
    (

```



```

        'partner_code' => $sPartnerCode,
        'language' => $sLanguageCode,
        'file' => 'skip_the_download_status.shtml',
        'tags' =>
        (
            'PERCENT_DISP' => $percent_disp,
            'PERCENT' => $percent,
            'FILE_NAME' => $filename,
            'FILE_SIZE' => $filesize,
            'TRANSFERRED' => $stransferred,
            'TRANSINFO' => $info,
            'K' => $k,
            'URL' => "/cgi-
bin/skip_the_download_status.cgi?seq=$nSeq&start=$start&pp=$percent&$sPartner
Params",
            'altURL' => $sAltURL
        )
    ));

    $template->clear;
    print "Content-type: text/html\n\n";
    print $template->get;
}

```

```
sub DisplayDone
```

```

{
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    &ErrorOut('skip_the_download_complete.shtml',
        $sFileURL,
        $sFileName,
        $sAltURL,
        $sPartnerCode,
        $sLanguageCode,
        $sMessage
    );
}

```

```
sub DisplayError
```

```

{
    my $sError = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    my $thtml = ($sAltURL != '')? 'skip_the_download_no_alt_error.shtml'
        : 'skip_the_download_error.shtml';

    &ErrorOut($thtml,
        $sFileURL,
        $sFileName,
        $sAltURL,
        $sPartnerCode,

```

```

        $sLanguageCode,
        $sError
    );
}

```

```

sub DisplayQuotaError
{
    my $sError = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    &ErrorOut('skip_the_download_quota_error.shtml',
        $sFileURL,
        $sFileName,
        $sAltURL,
        $sPartnerCode,
        $sLanguageCode,
        $sError
    );
}

```

```

sub ErrorOut ()
{
    my $sTHTMLFILE = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sMessage = shift;

    my $template = new XDrive::Template
    (
        (
            'language' => $sLanguageCode,
            'partner_code' => $sPartnerCode,
            'file' => $sTHTMLFILE,
            'tags' =>
            {
                'message' => $sMessage,
                'altURL' => $sAltURL,
                'fileURL' => $sFileURL,
                'FILE_NAME' => $sFileName,
                'LANG' => $sLanguageCode,
                'ALTURL' => $sAltURL,
                'STDPARTNER' => $sPartnerCode,
            }
        )
    );

    my $html = $template->get;

    print "Content-type: text/html\n\n";
    print $html;
}

```



```

        $sLanguageCode,
        $sError
    );
}

sub DisplayQuotaError
{
    my $sError = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    &ErrorOut('skip_the_download_quota_error.shtml',
        $sFileURL,
        $sFileName,
        $sAltURL,
        $sPartnerCode,
        $sLanguageCode,
        $sError
    );
}

```

```

sub ErrorOut ()
{
    my $sTHTMLFILE = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sMessage = shift;

    my $template = new XDrive::Template
    (
        'language' => $sLanguageCode,
        'partner_code' => $sPartnerCode,
        'file' => $sTHTMLFILE,
        'tags' =>
        (
            'message' => $sMessage,
            'altURL' => $sAltURL,
            'fileURL' => $sFileURL,
            'FILE_NAME' => $sFileName,
            'LANG' => $sLanguageCode,
            'ALTURL' => $sAltURL,
            'STDPARTNER' => $sPartnerCode,
        )
    );

    my $html = $template->get;

    print "Content-type: text/html\n\n";
    print $html;
}

```

```

my $sUser_name = $oUserInfo->fetchColumn('NAME_FIRST') . " " .
$oUserInfo->fetchColumn('NAME_LAST');
my $sUser_email = $oUserInfo->fetchColumn('EMAIL_ADDRESS');
$oUserInfo->finish();
$oUserInfo->disconnect();

if ($sAddress)
{
    &send_mail($sName, $sAddress, $sUser_name, $sUser_email,
    $nUser_ID, $oCGI, $oToken, $oErr, $oCookie);
    &display_thank_you($oCGI, $oCookie);
}
else
{
    &display_form($oCGI, $oCookie);
}
}

sub send_mail {
    my ($sName, $sAddress, $sUser_name, $sUser_email, $nUser_ID, $oCGI,
    $oToken, $oErr, $oCookie) = @_ ;

    ## send out email for each friend only if form is filled out
    ## get number of friend fields

    my $numFriends = $oCGI->param("numFriends");
    for (my $i=1; $i<=$numFriends; $i++)
    {
        $sAddress = $oCGI->param('friends_email' . $i);
        $sName = $oCGI->param('friends_name' . $i);
        my $sMessage = &get_message($sUser_name, $nUser_ID, $sName,
        $sUser_email, $oCookie);

        ##only send the mail if the email address is filled out
        if ($sAddress)
        {
            my %toXdrive =
            (
                To      => "$sName <$sAddress>",
                Bcc      => '',
                From      => "$sUser_email",
                Message   => $sMessage,
                Subject   => "Check out X:drive!",
            );

            unless (sendmail %toXdrive)
            {
                warn "## Mail error ".$Mail::Sendmail::error;
                if ($Mail::Sendmail::error =~ /451/)
                {
                    my $sError = $oErr->ReturnMessageGivenCode(1310);
                    XDErrorToBrowser("", $sError, undef, $oToken);
                }
            }
        }
        else
        {
            my $sError = $oErr->ReturnMessageGivenCode(1311);
            XDErrorToBrowser('tell_a_friend_error.shtml', $sError, undef, $oToken);
        }
        exit(1);
    }
}

```

```

    }

}

sub get_formfield {
    my ($sNum,$oCookie) = @_ ;

    my $oFormField = new XDrive::Template
        (
            'language'      => $oCookie->getElement('language'),
            'partner_code' => $oCookie->getElement('partner'),
        );
    $oFormField->load('tell_form_fields.shtml');

    $oFormField->tags
        (
            'number' => $sNum
        );

    return $oFormField->get;
}

sub get_message {
    my ($sUser_name, $nUser_ID, $sName, $sUserEmail,$oCookie) = @_ ;

    my $oMessage = new XDrive::Template
        (
            'language'      => $oCookie->getElement('language'),
            'partner_code' => $oCookie->getElement('partner'),
        );
    $oMessage->load('tell_a_friend_message.shtml');

    $oMessage->tags
        (
            'user_name' => $sUser_name,
            'nUser_ID' => $nUser_ID,
            'user_email' => $sUserEmail,
            'friend_name' => $sName
        );

    return $oMessage->get;
}

sub display_form {
    my $oCGI = shift;
    my $oCookie = shift;
    my $oForm = new XDrive::Template
        (
            'language'      => $oCookie->getElement('language'),
            'partner_code' => $oCookie->getElement('partner'),
        );
    $oForm->load('tell_a_friend.shtml');
    my $numFriends = $oCGI->param("numFriends");

    ##construct the html for multiple input fields
    my $inputFields='';

    for (my $i=1; $i<=$numFriends ; $i++)
    {
        $inputFields = $inputFields . &get_formfield($i,$oCookie);
    }
}

```

```
    }

    $oForm->tags
        (
            'friendsToTell' => $inputFields,
            'numFriends' => $numFriends,
        );
    print $oCGI->header, $oForm->get;
    exit(0);
}

sub display_thank_you {
    my $oCGI = shift;
    my $oCookie = shift;
    my $oForm = new XDrive::Template
        (
            'language' => $oCookie->getElement('language'),
            'partner_code' => $oCookie->getElement('partner'),
        );
    $oForm->load('tell_a_friend__t_y.thtml');
    print $oCGI->header, $oForm->get;
    exit(0);
}
```

###web_unauthorized.cgi

```
#!/usr/bin/perl
# Written by Martin Hald <mhald@uci.edu> on Sat Feb 13, 1999
#
# Program for showing unauthorized information and allowing the users to
# re-login and possibly showing them a "forgot your password?" link.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header param);
use CGI::Carp qw(fatalsToBrowser);
# use XDrive::CGI qw(:MAIN);
use XDrive::Client::Registration;
use XDrive::Template;
use XDrive::Error;

exit &main;

sub main
{
    my $oCGI = CGI->new();

    my $oLayout = new XDrive::Template;
    my $oContent = new XDrive::Template;
    my $oNavigation = new XDrive::Template;

    $oLayout->partner('xdrv');
    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');

    $oLayout->load('layout.thtml');
    $oNavigation->load('front_nav.thtml');

    ## Get the error key
    my $sError = $oCGI->param('error');

    ##now get the error message associated with that error
    my $oErr = new XDrive::Error;
    my $message = $oErr->ReturnMessageGivenCode($sError);

    ## Load the required template HTML files.
    my $oForm = new XDrive::Template;
    $oForm->partner('xdrv');
    $oForm->load("front_nav.thtml");
    $oContent->load("unauthorized.thtml");

    ## Update the layout
    $oLayout->tags
    (
        {
            'header_graphic' => 'header_denied.gif'
        }
    );

    ## Update the content
    $oContent->tags
    (
        {
            'error_message' => $message
        }
    );
    $oContent->clear();
}
```



```
## Print out the HTML and exit
$arLayout->tags
    (
        'content' => $oContent->get,
        'navigation' => $oNavigation->get,
        'title' => 'Authorization Denied'
    );
print header(), $arLayout->get;
return 0;
```

)

Windows Client Code

// Module: dlgShareAFile.h	1
// Module: dlgShareAFile.h	3
// Module: xdBase64.cpp	5
// Module: xdBase64.h	9
// Module: xdGlobals.h	10
// Module: xdParseDate.h.....	13
// Module: xdRegistry.h	14
// Module: xdTokens.h.....	16
// Module: xdTools.h.....	17
// Module: xdEngine.h	20
// Module: tdimsgtbl.h.....	22
// Module: tdisock.h	24
// Module: xdFileIO.cpp.....	41
// Module: xdDebugger.cpp	45

//

// **Module: dlgShareAFile.h**// Subsystem: KnoWare Internet Engine (kwEngine.dll)
// Contents: Declaration module for the dlgShareAFile class.

//

// -----
// Copyright (c) 1999 by X:drive(tm), Inc.
// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
// All rights reserved.
// -----
// -----
//#include "stdafx.h"
#include <xdGlobals.h>#ifndef _VXD_SOURCE_
#include "resource.h"
#endif
#include "dlgShareAFile.h"#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif// -----
// Implementation
// -----
BEGIN_MESSAGE_MAP(dlgShareAFile, CDialog)
 //{{AFX_MSG_MAP(dlgShareAFile)
 //}}AFX_MSG_MAP
END_MESSAGE_MAP()// -----
// Method: dlgShareAFile()
// Purpose: Standard constructor
//
dlgShareAFile::dlgShareAFile(CWnd* pParent /*=NULL*/) : CDialog(dlgShareAFile::IDD, pParent){
 {{{AFX_DATA_INIT(dlgShareAFile)
 m_sFileName = szEMPTY;
 m_sFileDescription = szEMPTY;
 m_sEmailMessage = szEMPTY;
 m_sEmailSubject = szEMPTY;
 m_sEmail0 = szEMPTY;
 m_sEmail1 = szEMPTY;
 m_sEmail2 = szEMPTY;
 m_sEmail3 = szEMPTY;
 m_sEmail4 = szEMPTY;
 }}}AFX_DATA_INIT
} // End of dlgShareAFile()// -----
// Method: DoDataExchange()
// Purpose: Standard data exchange handler
//
void dlgShareAFile::DoDataExchange(CDataExchange* pDX)
{

```

CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(dlgShareAFile)
DDX_Text(pDX, IDC_SHARE_FILENAME, m_sFileName);
DDX_Text(pDX, IDC_SHARE_FILEDESC, m_sFileDescription);
DDX_Text(pDX, IDC_SHARE_EMAILMSG, m_sEmailMessage);
DDX_Text(pDX, IDC_SHARE_EMAILSUB, m_sEmailSubject);
DDX_Text(pDX, IDC_SHARE_EMAIL1, m_sEmail0);
DDX_Text(pDX, IDC_SHARE_EMAIL2, m_sEmail1);
DDX_Text(pDX, IDC_SHARE_EMAIL3, m_sEmail2);
DDX_Text(pDX, IDC_SHARE_EMAIL4, m_sEmail3);
DDX_Text(pDX, IDC_SHARE_EMAIL5, m_sEmail4);
//}}AFX_DATA_MAP
} // End of DoDataExchange()

// -----
// Method: OnInitDialog()
// Purpose: Called to initialize the contents of the dialog
//
BOOL dlgShareAFile::OnInitDialog()
{
    CDialog::OnInitDialog();

    UpdateData(FALSE);
    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
} // End of OnInitDialog()

// -----
// Method: OnOK()
// Purpose: Called to close out the dialog.
//
void dlgShareAFile::OnOK()
{
    UpdateData(TRUE);
    CDialog::OnOK();
} // End of OnOK()

```

//

// Module: dlgShareAFile.h

// Subsystem: KnoWare Internet Engine (kwEngine.dll)
 // Contents: Declaration module for the dlgShareAFile class.

//

// -----
 // Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.

//

// -----
 //

#if !defined(INC_DLGSAREAFIIE_H_)
 #define _INC_DLGSAREAFIIE_H_

#if _MSC_VER > 1000
 #pragma once
 #endif // _MSC_VER > 1000

#ifndef _VXD_SOURCE_
 #include "resource.h"
 #endif

#ifndef _VXD_SOURCE_

// -----

// \ dlgShareAFile dialog class

//

class dlgShareAFile : public CDialog
 {
 public:

dlgShareAFile(CWnd* pParent = NULL); // standard constructor

//{AFX_DATA(dlgShareAFile)
 enum { IDD = IDD_SHARE };
 CString m_sFileName;
 CString m_sFileDescription;
 CString m_sEmailMessage;
 CString m_sEmailSubject;
 CString m_sEmail0;
 CString m_sEmail1;
 CString m_sEmail2;
 CString m_sEmail3;
 CString m_sEmail4;
 //}AFX_DATA

//{AFX_VIRTUAL(dlgShareAFile)
 protected:
 virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
 //}AFX_VIRTUAL

protected:

//{AFX_MSG(dlgShareAFile)
 virtual BOOL OnInitDialog();
 virtual void OnOK();
 //}AFX_MSG
 DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}

// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif

#endif // !defined(_INC_DLGSHAREAFILE_H)

//

// Module: xdBase64.cpp

// Subsystem: X:drive Client Engine (xdEngine.dll)
 // Contents: Implementation module for the xdBase64 class

//

// -----
 // Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.

//

// -----
 //

#include "stdafx.h"
 #include "xdBase64.h"

#ifdef _DEBUG
 #undef THIS_FILE
 static char THIS_FILE[] = __FILE__;
 #endif
 #ifdef _VXD_SOURCE_
 #include <xdEngine.h>
 #define TRACE DEBUG_DPRINTF
 #endif

// Static Member Initializers
 //

// The 7-bit alphabet used to encode binary information
 CString xdBase64::m_sBase64Alphabet =
 _T("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/");

int xdBase64::m_nMask[] = { 0, 1, 3, 7, 15, 31, 63, 127, 255 };

// -----

// Method: xdBase64()
 // Purpose: Standard Constructor
 //
 xdBase64::xdBase64 (void)
 {
 } // End of xdBase64()

// -----

// Method: ~xdBase64()
 // Purpose: Standard destructor
 //
 xdBase64::~~xdBase64()
 {
 } // End of ~xdBase64()

// -----

// Method: Encode()
 // Purpose: Encodes a string
 //
 CString xdBase64::Encode(LPCTSTR szEncoding, int nSize)
 {

 CString sOutput = _T("");
 int nNumBits = 6;
 UINT nDigit;
 int lp = 0;

```

    ASSERT( szEncoding != NULL );
    if( szEncoding == NULL )
        return sOutput;
    m_szInput = szEncoding;
    m_nInputSize = nSize;

    m_nBitsRemaining = 0;
    nDigit = read_bits( nNumBits, &nNumBits, lp );
    while( nNumBits > 0 )
    {
        sOutput += m_sBase64Alphabet[ (int)nDigit ];
        nDigit = read_bits( nNumBits, &nNumBits, lp );
    }
    // Pad with '=' as per RFC 1521
    while( sOutput.GetLength() % 4 != 0 )
    {
        sOutput += '=';
    }
    return sOutput;
} // End of Encode()

// -----
// Method: Decode()
// Purpose: Decodes data
// Notes: The size of the output buffer must not be less than 3/4 the
//        size of the input buffer. For simplicity, make them the same
//        size.
//
int xdBase64::Decode(LPCTSTR szDecoding, LPTSTR szOutput)
{
    CString sInput;
    int c, lp = 0;
    int nDigit;
    CString strDecode;
    int* pDecode = (int*)strDecode.GetBuffer(256*sizeof(int));

    ASSERT( szDecoding != NULL );
    ASSERT( szOutput != NULL );
    if( szOutput == NULL )
        return 0;
    if( szDecoding == NULL )
        return 0;
    sInput = szDecoding;
    if( sInput.GetLength() == 0 )
        return 0;

    // Build Decode Table
    //
    for( int i = 0; i < 256; i++ )
        pDecode[i] = -2; // Illegal digit
    for( i = 0; i < 64; i++ )
    {
        pDecode[ m_sBase64Alphabet[ i ] ] = i;
        pDecode[ m_sBase64Alphabet[ i ] | 0x80 ] = i; // Ignore 8th bit
        pDecode[ '=' ] = -1;
        pDecode[ '=' | 0x80 ] = -1; // Ignore MIME padding char
    }

    // Clear the output buffer
    memset( szOutput, 0, sInput.GetLength() + 1 );

    // Decode the Input

```



```

//
for( lp = 0, i = 0; lp < sInput.GetLength(); lp++ )
{
    c = sInput[ lp ];
    nDigit = pDecode[ c & 0x7F ];
    if( nDigit < -1 )
    {
        return 0;
    }
    else if( nDigit >= 0 )
        // i (index into output) is incremented by write_bits()
        write_bits( nDigit & 0x3F, 6, szOutput, i );
}

return i;
} // End of Decode()

// -----
// Method: read_bits()
// Purpose: dunno
//
UINT xdBase64::read_bits(int nNumBits, int * pBitsRead, int& lp)
{
    ULONG lScratch;
    while( ( m_nBitsRemaining < nNumBits ) &&
           ( lp < m_nInputSize ) )
    {
        int c = m_szInput[ lp++ ];
        m_lBitStorage <<= 8;
        m_lBitStorage |= (c & 0xff);
        m_nBitsRemaining += 8;
    }
    if( m_nBitsRemaining < nNumBits )
    {
        lScratch = m_lBitStorage << ( nNumBits - m_nBitsRemaining );
        *pBitsRead = m_nBitsRemaining;
        m_nBitsRemaining = 0;
    }
    else
    {
        lScratch = m_lBitStorage >> ( m_nBitsRemaining - nNumBits );
        *pBitsRead = nNumBits;
        m_nBitsRemaining -= nNumBits;
    }
    return (UINT)lScratch & m_nMask[nNumBits];
} // End of read_bits()

// -----
// Method: write_bits()
// Purpose: dunno
//
void xdBase64::write_bits ( UINT nBits, int nNumBits, LPTSTR szOutput, int& i )
{
    UINT nScratch;

    m_lBitStorage = (m_lBitStorage << nNumBits) | nBits;
    m_nBitsRemaining += nNumBits;
    while( m_nBitsRemaining > 7 )
    {
        nScratch = m_lBitStorage >> (m_nBitsRemaining - 8);
        szOutput[ i++ ] = (TCHAR)(nScratch & 0xFF);
        m_nBitsRemaining -= 8;
    }
}

```

WO 01/33381
} // End of write_bits()

PCT/US00/30536

```

//
// Module: xdBase64.h
// Subsystem: X:drive Client Engine (xdEngine.dll)
// Contents: Declaration module for the xdBase64 class.
//
// -----
// Copyright (c) 1999 by X:drive(tm), Inc.
// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
// All rights reserved.
// -----
//
// #if !defined( _INC_XDBASE64_H )
// #define _INC_XDBASE64_H_

// #ifdef _VXD_SOURCE_
// #include <xdCString.h>
// #endif

// #if _MSC_VER >= 1000
// #pragma once
// #endif // _MSC_VER >= 1000

// -----
// xdBase64 encoder class
//
class xdBase64
{
public:
    xdBase64 ( void );
    virtual ~xdBase64 ( void );

    virtual int          Decode ( LPCTSTR szDecoding, LPTSTR szOutput );
    virtual CString Encode ( LPCTSTR szEncoding, int nSize );

protected:
    void          write_bits ( UINT nBits, int nNumBts, LPTSTR szOutput, int& lp );
    UINT          read_bits ( int nNumBits, int* pBitsRead, int& lp );

protected:
    int          m_nInputSize;
    int          m_nBitsRemaining;
    ULONG        m_lBitStorage;
    LPCTSTR      m_szInput;
    static int m_nMask[];
    static CString m_sBase64Alphabet;
};

// #endif // !defined( _INC_XDBASE64_H )

```

// Module: xdGlobals.h

// Subsystem: X:drive
 // Contents: Global definitions used throughout the system
 //

// -----
 // Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.
 // -----
 // -----
 //

#ifndef _INC_XDGLOBALES_H_
 #define _INC_XDGLOBALES_H_

#ifdef _VXD_SOURCE_

//
 // This HodgePodge helps us to be able to compile all of our code
 // under Ring-3 and Ring-0 without too much modification.
 //

#ifndef USE_NDIS
 #define USE_NDIS
 #endif

#include <vtoolscp.h> // VToolsD main header file

#ifndef LPCTSTR
 typedef char TCHAR;
 typedef unsigned char TUCCHAR;
 typedef const TCHAR* LPCTSTR;
 typedef TCHAR* LPTSTR;
 typedef unsigned char BYTE;
 typedef BYTE* LPBYTE;
 typedef DSKTLSYSTEMTIME SYSTEMTIME;
 typedef HANDLE HINSTANCE;
 #define _T(x) (x)
 #endif

#ifndef BASED_CODE
 #define BASED_CODE
 #endif

#ifndef INVALID_HANDLE_VALUE
 #define INVALID_HANDLE_VALUE (HANDLE)-1
 #endif

#define _tcsstr strstr // Standard unicode mappings
 #define _tcslen strlen
 #define _tcsncpy strcpy
 #define _tcsrchr strchr
 #define _tcscat strcat
 #define _ttoi atoi
 #define _ttol atol
 #define _tcsrev strrev
 #define _tcschr strchr
 #define _tcsncpy strncpy
 #define _tcsprbrk strprbrk
 #define _stprintf sprintf
 #define _tcslwr strlwr

```

#define _tcsupr      strupr
#define _tcsicmp     stricmp
#define _tcscmp      strcmp
#define _tcscoll     strcmp
#define _istdigit    isdigit
// #define ASSERT Assert
typedef HANDLE      HWND;
#endif

// -----
// Setup a whole bunch of constants that we can use throughout the systems
//
#define chNL          _T("\n")
#define chCOMMA       _T(',')
#define chDOSSLASH    _T("\")
#define chUNIXSLASH   _T('/')
#define chQUOTE       _T("\"")
#define chDQUOTE      _T("\"")
#define chPERIOD      _T('.')
#define chBAR         _T('|')
#define chTAB         _T('\t')
#define chCR          _T('\r')
#define chSPACE       _T(' ')
#define chCOLON       _T(':')
#define chSEMICOLON   _T(';')
#define chDASH        _T('-')
#define chPLUS        _T('+')
#define chPERCENT     _T('%')
#define chOPENBRACKET _T('[')
#define chCLOSEBRACKET _T(']')
#define chNUL         _T('\0')
#define chZERO        _T('0')
#define chONE         _T('1')
#define chTWO         _T('2')
#define chTHREE       _T('3')
#define chFOUR        _T('4')
#define chFIVE        _T('5')
#define chSIX         _T('6')
#define chSEVEN       _T('7')
#define chEIGHT       _T('8')
#define chNINE        _T('9')
#define chOPENPAREN   _T('(')
#define chCLOSEPAREN _T(')')
#define chAT          _T('@')

#define szNL          _T("\n")
#define szCOMMA       _T(",")
#define szDOSSLASH    _T("\")
#define szUNIXSLASH   _T("/")
#define szQUOTE       _T("\"")
#define szDQUOTE      _T("\"")
#define szPERIOD      _T(".")
#define szBAR         _T("|")
#define szTAB         _T("\t")
#define szCR          _T("\r")
#define szSPACE       _T(" ")
#define szCOLON       _T(":")
#define szSEMICOLON   _T(";")
#define szDASH        _T("-")
#define szPLUS        _T("+")
#define szOPENBRACKET _T("[")
#define szCLOSEBRACKET _T("]")

```

```

#define szAT _T("@")
#define szEMPTY _T("")
#define szCURRENTDIR _T(".")
#define szPARENTDIR _T("..")
#define szFTP_DOT _T("ftp.")
#define szFTP_SLASH _T("ftp://")
#define szOPENPAREN _T("(")
#define szCLOSEPAREN _T(")")

#define XD_CACHE_BASEDIR _T("xdcache")
#define XD_LOGFILE_NP _T("xdrive.log")
#define XD_LOGFILE_VXD _T("xdrivevxd.log")

// -----
// We need to define the scope of values which will be used in the system.
// They are defined here since we need to read/write these to the registry.
//

//
// General defines
//
#define XD_LEN_32 32
#define XD_LEN_64 64
#define XD_LEN_128 128
#define XD_LEN_256 256
#define XD_LEN_512 512
#define XD_LEN_1024 1024
#define XD_LEN_2048 2048

//
// these program IDs are also the 1st two digits of the registration number
//
#define XD_PROGID_XDRIVE 0x53 // {DB2112AD-0000-0000-0053-000004281965}

//
// IN will generate a directory listing and the local file that contains
// that information will have an extension of '.fnd'. For example, if
// IN/FND does a directory listing of ftp.microsoft.com/softlib/mslfiles,
// it will place the raw directory listing in the in the local IN cache
// directory (which is currently defined as hanging off of the same
// directory where IN is located) as
//
// c:\xdCache\ftp.microsoft.com\root.softlib.mslfiles.ls
//
// and the parsed FND formatted data will be placed into
//
// c:\xdCache\ftp.microsoft.com\root.softlib.mslfiles.fnd
//
// the .fnd file is parsed out to produce the information returned as a
// result of the FINDFIRST()/FINDNEXT() calls to the NP.
//
#define XD_FILEEXT_LS _T(".ls")
#define XD_FILEEXT_XDR _T(".fnd")
//
// Here is our Network Provider Name
//
#define XD_PROVIDER_NAME _T("Xdrive")
#define XD_PROVIDER_NETID 0x00120000

#endif // _INC_XDGLOBALS_H_

```

//

// **Module: xdParseDate.h**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Declaration module for the CParseDate utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnowWare(r), Inc.

// All rights reserved.

// -----

// -----

//

#ifndef _INC_XDPARSEDAT_H_

#define _INC_XDPARSEDAT_H_

#include <xdTokens.h>

class XDTOOLS_PUBLIC CParseDate

{

public:

CParseDate (void);

~CParseDate (void);

BOOL Parse (LPCTSTR s);

int	m_iYear;
int	m_iMonth;
int	m_iDay;
int	m_iHour;
int	m_iMinute;
int	m_iSecond;
TCHAR	m_szDate[64];
TCHAR	m_szTime[32];
TCHAR	m_szOrig[64];

private:

BOOL isNUM (LPCTSTR s);

BOOL isDOW (LPCTSTR s);

xdTokens m_tokens;

};

#endif

//

// **Module: xdRegistry.h**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Declaration module for the xdRegistry utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

// -----

// -----

//

#ifndef _INC_XDREGISTRY_H_

#define _INC_XDREGISTRY_H_

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

#include <xdGlobals.h>

// X:drive system wide globals

#include <xdTools.h>

// X:drive Tools Related

// -----

// xdRegistry

// the registry class encapsulates the registry functions. You must open

// at least a hive in the constructor. then you can optionally open

// a subkey & read/write information to the registry. All methods will return

// true upon successful completion. false will be returned if an error

// has occurred.

//

class XDTOOLS_PUBLIC xdRegistry

{

public:

xdRegistry();

~xdRegistry();

//

// public interface

//

public:

BOOL RegOpenRead (HKEY hHive, LPCTSTR szSubKey);

BOOL RegOpenWrite (HKEY hHive, LPCTSTR szSubKey);

BOOL RegClose (void);

BOOL RegDeleteKey (HKEY hHive, LPCTSTR szSubKey);

BOOL RegDeleteValue (LPCTSTR szVal);

BOOL RegEnumKey (int i, LPCTSTR szKeyName, UINT uiLenWithNull);

BOOL RegEnumVal (int i, LPCTSTR szValName, UINT uiLenWithNull, LPCTSTR

szValData, UINT uiDataLenWithNull);

BOOL RegEnumStr (int i, LPCTSTR szVal, UINT uiLenWithNull);

BOOL RegGetStr (LPCTSTR sName, LPCTSTR szVal, UINT uiLenWithNull);

BOOL RegPutStr (LPCTSTR sName, LPCTSTR szVal);

BOOL RegPutBin (LPCTSTR sName, BYTE* pBuffer, UINT uiLen);

BOOL RegGetNum (LPCTSTR sName, BOOL& bVal);

BOOL RegGetNum (LPCTSTR sName, WORD& wVal);

BOOL RegGetNum (LPCTSTR sName, DWORD& dwVal);

BOOL RegGetNum (LPCTSTR sName, UINT& uiVal);

BOOL RegPutNum (LPCTSTR sName, DWORD dwVal);


```
LONG      RegGetLastError ( void );

private:
    HKEY     m_hKey;      // the current open hive
    LONG     m_lRetCode;  // the last return code
}; // End of xdRegistry

#endif // _INC_XDREGISTRY_H_
```

//

// **Module: xdTokens.h**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Declaration module for xdTokens utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDTOKENS_H_

#define _INC_XDTOKENS_H_

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h> // X:drive Tools Related

#define XD_MAX_TOKENS 1024

// -----

// xdTokens

// This class is a big worker class. its used to parse strings into

// tokens or substrings. Strings are parsed by supplying a string of

// characters which will be used to parse out the string.

//

class XDTOOLS_PUBLIC xdTokens

{

public:

xdTokens(LPCTSTR pTokens = NULL);

~xdTokens();

//

// Public Interface

//

public:

int Parse(int iNumToParse, LPCTSTR pString, LPCTSTR pTokens=NULL);

int Parse(LPCTSTR pString, LPCTSTR pTokens=NULL);

LPCTSTR operator[](int iIndex);

//

// Private Members

//

private:

LPCTSTR

*m_pTok;

int

m_iNumParsed;

LPTSTR

m_szWorkString;

LPTSTR

m_szTokens;

LPTSTR

m_pWorkString;

}; // End of xdTokens

#endif // _INC_XDTOKENS_H_

//

// Module: xdTools.h

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Main header file for the xdTools library

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDTOOLS_H_

#define _INC_XDTOOLS_H_

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

#include <xdGlobals.h>

// X:drive system wide globals

#ifdef _VXD_SOURCE_

#include <xdCString.h>

#endif

#pragma warning (disable : 4100)

#pragma warning (disable : 4201)

//

// The following code block will insure the proper resolution of any

// API functions (and classes) which are exposed from the XDTOOLS library.

// When compiling the XDTOOLS library source code, make sure that the

// following #define is defined in the project settings (both debug & release).

// This will cause any classes and/or API functions defined as to

// be exported to the LIB file. If you are USING the library by linking to

// the XDTOOLS.LIB or XDTOOLSD.LIB import libraries, then ignore the

// following #define's for

//

#ifdef _XDTOOLS_SOURCE_

#define XDTOOLS_PUBLIC __declspec(dllexport)

#else

#define XDTOOLS_PUBLIC // __declspec(dllimport)

#endif // _XDTOOLS_SOURCE_

//

// If we are debugging & we trap an exception, we will display it

// in a message box, otherwise in release mode, we wont.

//

#ifdef _DEBUG

#define XDTRACE(x) AfxMessageBox(x)

#else

#define XDTRACE(x) TRACE0(x)

#endif

// -----

// XDDATE API (Date Functions)

//

XDTOOLS_PUBLIC int XDDATE_MonthNum (LPTSTR szMonth);

// -----

// XDSTR API (String Functions)

//

```

XDTOOLS_PUBLIC LPTSTR XDSTR_Squish ( LPTSTR p );
XDTOOLS_PUBLIC LPTSTR XDSTR_StripChar ( LPTSTR p, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_DirSlashAdd ( LPTSTR sz, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_DirSlashRemove ( LPTSTR sz, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_TrimRight ( LPTSTR );
XDTOOLS_PUBLIC LPTSTR XDSTR_TrimLeft ( LPTSTR );
XDTOOLS_PUBLIC LPTSTR XDSTR_Trim ( LPTSTR );
XDTOOLS_PUBLIC BOOL XDAPI_CreatePath ( LPCTSTR ); // calls CreateDirectory() to make a path.

```

```

// -----
// Stuff for message boxes
//
#ifdef _VXD_SOURCE_
    int XDTOOLS_PUBLIC XD_MSG ( LPCTSTR szText, UINT uiMsgFlags );
    int XDTOOLS_PUBLIC XD_QUESTION ( LPCTSTR szText, UINT uiMsgFlags );
    LPCTSTR XDTOOLS_PUBLIC XD_TEXT ( HINSTANCE h, UINT uiResId ); // LOADS A
RESOURCE!
    BOOL XD_DoHelp ( LPHELPINFO );
    void XD_DoHelpContext ( CWnd* );
#endif

```

```

//
// the calling object needs to supply the resource
// handle for loading the string. So set up a stupid macro
// that will automatically supply this!
//
#define XD_LOADSTRING(x) XD_TEXT(AfxGetResourceHandle(),(x))

```

```

//
// DEBUGGING STUFF
//
#define CATCH_MSG_T("Caught Exception in File %s, Line %d\n\n")
#ifdef _VXD_SOURCE_
    #define XDCATCH dprintf(CATCH_MSG, T(_FILE_), _LINE_)
#else
    #define XDCATCH { CString s; s.Format(CATCH_MSG, T(_FILE_), _LINE_);
AfxMessageBox(s); }
#endif

```

```

//
// Ring 0 File I/O
//
#ifdef _VXD_SOURCE_
#define GENERIC_READ (0x80000000) /* from WINNT.H */
#define GENERIC_WRITE (0x40000000) /* from WINNT.H */
#define CREATE_NEW 1
#define CREATE_ALWAYS 2
#define OPEN_EXISTING 3
#define OPEN_ALWAYS 4
#define TRUNCATE_EXISTING 5
#define FILE_SHARE_READ 0x00000001
#define FILE_SHARE_WRITE 0x00000002
#define FILE_SHARE_DELETE 0x00000004 // not supported

```

```

HANDLE CreateFile ( LPCTSTR lpFileName, // pointer to name of the file
DWORD dwDesiredAccess, // access (read-write) mode
DWORD dwShareMode, // share mode
void* lpSecAtt, // pointer to security
attributes
DWORD dwCreateFlags, // how to create
DWORD dwFlagsAndAttributes, // file attributes
HANDLE);

```

```

BOOL CloseHandle ( HANDLE hFile );
BOOL ReadFile ( HANDLE hFile,          // handle of file to read
               void* lpBuffer,         // pointer to buffer that receives data
               DWORD nNumberOfBytesToRead, // number of bytes to read
               DWORD* lpNumberOfBytesRead, // pointer to number of bytes read
               void* lpOverlapped);    // pointer to structure for data

BOOL ReadFileLine ( HANDLE hFile,          // handle of file to read
                   data BYTE* lpBuffer,    // pointer to buffer that receives
                                // number of bytes to read
                                // pointer to number of bytes read
                                // pointer to structure for data
                   DWORD dwBytesToRead,
                   DWORD* dwBytesRead,
                   DWORD* dwOffset);

BOOL WriteFile ( HANDLE hFile, LPCTSTR lpBuffer, DWORD dwBytesToWrite,
                DWORD* pBytesWritten, void* p);

DWORD GetFileSize ( HANDLE hFile, DWORD* pdwHigh );
#endif
#endif // !defined(_INC_XDTOOLS_H_)

```

//

// Module: xdEngine.h

// Subsystem: X:drive Client Engine (xdEngine.dll)
 // Contents: Main include file for the xdEngine subsystem
 //

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.

//

// -----

//

#ifndef _INC_XDRIVE_ENGINE_H_
 #define _INC_XDRIVE_ENGINE_H_

#if _MSC_VER >= 1000
 #pragma once
 #endif // _MSC_VER >= 1000

#pragma warning (disable : 4100)
 #pragma warning (disable : 4201)

#ifdef _XDENGINE_SOURCE_
 #define XDAPI_PUBLIC __declspec(dllexport)
 #else
 #define XDAPI_PUBLIC // __declspec(dllimport)
 #endif // _XDENGINE_SOURCE_

#pragma pack(1) // byte pack this thing!

#include <xdGlobals.h>

// -----

// XD_DIRENTRY - directory listing item

//

// The following structure is used to hold an object in the file listing
 // file. Xdrive will generate the file list for the directory and store it
 // in the cache directory. That file will contain
 // a list of record structures of this type. The .mnd file is generated
 // based upon the FTP server specific format in the .idx file in the same
 // cache directory.

//

typedef struct _xd_direntry_
 {

 USHORT cb; // class size, MUST BE FIRST!!!!
 DWORD dwFileAttributes;
 FILETIME ftCreationTime;
 FILETIME ftLastAccessTime;
 FILETIME ftLastWriteTime;
 DWORD nFileSizeHigh;
 DWORD nFileSizeLow;
 TCHAR cFileName[XD_LEN_512];
 TCHAR m_szObPerms [XD_LEN_32 + 1];
 BYTE m_bObOwnerPerms[4];
 BYTE m_bObGroupPerms[4];
 BYTE m_bObWorldPerms[4];
} XD_DIRENTRY, *LPXD_DIRENTRY;

#pragma pack()

```

//
// Return codes
//
typedef UINT    XD_RETCODE;

#define XD_SUCCESS                (int)0
#define XD_CANCEL                 (int)1
#define XD_ERR_CONNECTFAILED     (int)2    // socket connect failed
#define XD_ERR_LOGINFAILED       (int)3    // bad username/pwd
#define XD_ERR_CONNECTREFUSED    (int)5    // socket connect refused
#define XD_ERR_CANTRESOLVEHOST   (int)6    // cant resolve host
#define XD_ERR_SERVERUPGRADING   (int)7    // upgrading our servers

#define XD_ERR_OTHER              (int)-1

//
// The following constants are used in the notification structure.
//
typedef enum
{
    XD_NOTIFY_IDLE                = 0,      // nothing happening here
    XD_NOTIFY_STATUS_MSG          = 1000,   // status msg
    XD_NOTIFY_XFERDATA_DN         = 1001,   // downloading
    XD_NOTIFY_XFERDATA_UP         = 1002,   // uploading
    XD_NOTIFY_QUOTA                = 1003,   // Update the quota
    XD_NOTIFY_START               = 1004,   // Start an operation
    XD_NOTIFY_STOP                = 1005    // Stop an operation
} XD_NOTIFY_CODE;

// -----
// XD_NOTIFY - This is our notification structure. The http engine
// will use this structure to pass status information back to the
// invoking method.
//
#pragma pack(1)

typedef struct _xd_notification_
{
    int                m_iNotifyType;
    TCHAR              m_szMessage [ 1024 + sizeof(TCHAR) ];

    //
    // used for send/receive
    //
    ULONG              m_dwStartTime;        // GetTickCount()/1000
    ULONG              m_dwCurrentTime;      // GetTickCount()/1000

    DWORD              m_dwCurrentBytes;
    DWORD              m_dwTotalBytes;

    TCHAR              m_szLocalFileName [ MAX_PATH + sizeof(TCHAR) ];
    TCHAR              m_szRemoteFileName [ MAX_PATH + sizeof(TCHAR) ];
} XD_NOTIFY, *LPXD_NOTIFY;

#pragma pack()
#define XD_NOTIFY_MAX      50

#endif // _INC_XDRIVE_ENGINE_H_

```

//

// **Module: tdimsgtbl.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: TDI Error table.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by River Front Software

// All rights reserved.

//

// -----

//

#ifndef __TDIMSGTBL_H

#define __TDIMSGTBL_H

typedef struct

```
{
    TDI_STATUS    Status;
    int           WinStatus;
    char          *szMsg;
} INETTDIMSG;
```

INETTDIMSG TdiMsgTbl[] =

```
{
    {TDI_SUCCESS, ERROR_SUCCESS, "TDI Success"},
    {TDI_NO_RESOURCES, ERROR_BAD_COMMAND, "No resources."},
    {TDI_ADDR_IN_USE, ERROR_BAD_COMMAND, "Address already in use."},
    {TDI_BAD_ADDR, ERROR_BAD_COMMAND, "Address given is bad."},
    {TDI_NO_FREE_ADDR, ERROR_BAD_COMMAND, "No addresses available."},
    {TDI_ADDR_INVALID, ERROR_BAD_COMMAND, "Address object is invalid."},
    {TDI_ADDR_DELETED, ERROR_BAD_COMMAND, "Address object was deleted."},
    {TDI_BUFFER_OVERFLOW, ERROR_BAD_COMMAND, "Buffer overflowed."},
    {TDI_BAD_EVENT_TYPE, ERROR_BAD_COMMAND, "Bad event type."},
    {TDI_BAD_OPTION, ERROR_BAD_COMMAND, "Bad option or length."},
    {TDI_CONN_REFUSED, ERROR_BAD_COMMAND, "Connection was refused."},
    {TDI_INVALID_CONNECTION, ERROR_BAD_COMMAND, "Invalid connection."},
    {TDI_ALREADY_ASSOCIATED, ERROR_BAD_COMMAND, "Connection already associated."},
    {TDI_NOT_ASSOCIATED, ERROR_BAD_COMMAND, "Connection not associated."},
    {TDI_CONNECTION_ACTIVE, ERROR_BAD_COMMAND, "Connection is still active."},
    {TDI_CONNECTION_ABORTED, ERROR_BAD_COMMAND, "Connection was aborted."},
    {TDI_CONNECTION_RESET, ERROR_BAD_COMMAND, "Connection was reset."},
    {TDI_TIMED_OUT, ERROR_BAD_COMMAND, "Connection timed out."},
    {TDI_GRACEFUL_DISC, ERROR_BAD_COMMAND, "Received a graceful disconnect."},
    {TDI_NOT_ACCEPTED, ERROR_BAD_COMMAND, "Data not accepted."},
    {TDI_MORE_PROCESSING, ERROR_BAD_COMMAND, "More processing required."},
    {TDI_INVALID_STATE, ERROR_BAD_COMMAND, "TCB in an invalid state."},
    {TDI_INVALID_PARAMETER, ERROR_BAD_COMMAND, "An invalid parameter."},
    {TDI_DEST_NET_UNREACH, ERROR_BAD_COMMAND, "Destination net is unreachable."},
    {TDI_DEST_HOST_UNREACH, ERROR_BAD_COMMAND, "Dest. host is unreachable."},
    {TDI_DEST_UNREACHABLE, ERROR_BAD_COMMAND, "Dest. is unreachable."},
    {TDI_DEST_PROT_UNREACH, ERROR_BAD_COMMAND, "Destination protocol is unreachable."},
    {TDI_DEST_PORT_UNREACH, ERROR_BAD_COMMAND, "Dest. port is unreachable."},
    {TDI_INVALID_QUERY, ERROR_BAD_COMMAND, "Invalid query type specified."},
    {TDI_REQ_ABORTED, ERROR_BAD_COMMAND, "Request was aborted for some reason."},
    {TDI_BUFFER_TOO_SMALL, ERROR_BAD_COMMAND, "Buffer was too small."},
    {TDI_CANCELLED, ERROR_BAD_COMMAND, "The request was cancelled."},
    {TDI_BUFFER_TOO_BIG, ERROR_BAD_COMMAND, "Invalid request."},
    {ERROR_SEM_TIMEOUT, ERROR_SEM_TIMEOUT, "Timed out."},
    {TDI_PENDING, ERROR_BAD_COMMAND, "Pending"}
}
```


};

#endif

//

// **Module: tdisock.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: TDI Socket header file.

//

// -----
// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by River Front Software

// All rights reserved.

// -----

// -----

//

#ifndef __TDISOCK_H

#define __TDISOCK_H

#define TDISOCK_TIMEOUT 15000

#define WSADESCRIPTION_LEN 256

#define WSASYS_STATUS_LEN 128

typedef short SHORT;

typedef unsigned short USHORT;

typedef unsigned short ushort;

typedef unsigned int uint;

typedef unsigned long ulong;

typedef unsigned long ULONG;

typedef void (*CTEReqCmpltRtn)(void *Context, long FinalStatus, unsigned int ByteCount);

typedef unsigned char uchar;

typedef struct WSADATA {

WORD wVersion;

WORD wHighVersion;

char szDescription[WSADESCRIPTION_LEN+1];

char szSystemStatus[WSASYS_STATUS_LEN+1];

unsigned short iMaxSockets;

unsigned short iMaxUdpDg;

char FAR * lpVendorInfo;

} WSADATA;

typedef WSADATA FAR *LPWSADATA;

#define USE_NDIS 1

#include <vtoolscp.h>

#include <ctrl.h>

#undef USE_NDIS

#include <tdi.h>

#include <vxdsvc.h>

#include <tdivxd.h>

#include <tdistat.h>

#undef VTDI_Device_ID

#include <vtdi.h>

#define MAKELONG(a, b) ((LONG)((((WORD)(a)) | ((DWORD)((WORD)(b))) << 16))

#define LOWORD(l) ((WORD)(l))

```

#define HIWORD(l)      (((WORD)(((DWORD)(l) >> 16) & 0xFFFF))
#define LOBYTE(w)      ((BYTE)(w))
#define HIBYTE(w)      ((BYTE)(((WORD)(w) >> 8) & 0xFF))

/*
 * Structures returned by network data base library, taken from the
 * BSD file netdb.h. All addresses are supplied in host order, and
 * returned in network order (suitable for use in system calls).
 */

struct hostent {
    char FAR * h_name;      /* official name of host */
    char FAR * FAR * h_aliases; /* alias list */
    short h_addrtype;      /* host address type */
    short h_length;        /* length of address */
    char FAR * FAR * h_addr_list; /* list of addresses */
#define h_addr h_addr_list[0] /* address, for backward compat */
};

/***** Wait for semaphore flags */
#define WAIT_SEMA_FLAGS 0 //BLOCK_SVC_INTS | BLOCK_POLL

/***** Macro to call wait on semaphore function */
#define SEMAPHORE_WAIT( hSem, nTimeout ) \
    WaitOnSemaphore( s, hSem, #hSem, nTimeout )

/***** Checks for valid TDI status */
#define TDI_CHECKSTATUS(s) if ( (s) != TDI_SUCCESS ) \
    { \
        \
        errdebug( DBG_log("ERROR - File: %s \
        \
        Line:%d TDI [%d] - %s\n", \
        \
        MapTdiToString(s)); ); \
        \
        goto Exit; \
    }

/***** Destroys a semaphore */
#define SEMAPHORE_SAFE_DESTROY(hSem) \
    if (hSem) \
    { \
        vbsdebug( DBG_log("Destroy Semaphore %s", #hSem); ); \
        UtilSemDestroy(hSem); \
        hSem = 0; \
    }

/***** Signals a semaphore */
#define SEMAPHORE_SAFE_SIGNAL(hSem) \
    if (hSem) \
    { \
        vbsdebug( DBG_log("**** Signal Semaphore %s", #hSem); ); \
        vbsdebug( DBG_log_hex_long( hSem ); ); \
        Signal_Semaphore_No_Switch( hSem ); \
    } \
    else \
    { \
        vbsdebug( DBG_log("**** NO SEMAPHORE TO SIGNAL %s", #hSem); ); \
    }

```

```

/*
 * Basic system type definitions, taken from the BSD file sys/types.h.
 */
typedef unsigned char  u_char;
typedef unsigned short u_short;
typedef unsigned int    u_int;
typedef unsigned long   u_long;

/*
 * Constants and structures defined by the internet system,
 * Per RFC 790, September 1981, taken from the BSD file netinet/in.h.
 */

/*
 * Protocols
 */
#define IPPROTO_IP      0      /* dummy for IP */
#define IPPROTO_ICMP    1      /* control message protocol */
#define IPPROTO_IGMP    2      /* internet group management protocol */
#define IPPROTO_GGP     3      /* gateway^2 (deprecated) */
#define IPPROTO_TCP     6      /* tcp */
#define IPPROTO_PUP     12     /* pup */
#define IPPROTO_UDP     17     /* user datagram protocol */
#define IPPROTO_IDP     22     /* xns idp */
#define IPPROTO_ND      77     /* UNOFFICIAL net disk proto */

#define IPPROTO_RAW     255     /* raw IP packet */
#define IPPROTO_MAX     256

/*
 * Port/socket numbers: network standard functions
 */
#define IPPORT_ECHO      7
#define IPPORT_DISCARD  9
#define IPPORT_SYSTAT   11
#define IPPORT_DAYTIME  13
#define IPPORT_NETSTAT  15
#define IPPORT_FTP       21
#define IPPORT_TELNET    23
#define IPPORT_SMTP      25
#define IPPORT_TIMESERVER 37
#define IPPORT_NAMESERVER 42
#define IPPORT_WHOIS     43
#define IPPORT_MTP       57

/*
 * Port/socket numbers: host specific functions
 */
#define IPPORT_TFTP      69
#define IPPORT_RJE       77
#define IPPORT_FINGER    79
#define IPPORT_TTYLINK  87
#define IPPORT_SUPDUP    95

/*
 * UNIX TCP sockets
 */
#define IPPORT_EXECSERVER 512

```

```

#define IPPORT_LOGINSERVER 513
#define IPPORT_CMDSERVER 514
#define IPPORT_EFSSERVER 520

/*
 * UNIX UDP sockets
 */
#define IPPORT_BIFFUDP 512
#define IPPORT_WHOSERVER 513
#define IPPORT_ROUTESEVER 520
/* 520+1 also used */

/*
 * Ports < IPPORT_RESERVED are reserved for
 * privileged processes (e.g. root).
 */
#define IPPORT_RESERVED 1024

/*
 * Link numbers
 */
#define IMPLINK_IP 155
#define IMPLINK_LOWEXPER 156
#define IMPLINK_HIGHEXPER 158

/*
 * Internet address (old style... should be updated)
 */
struct in_addr {
    union {
        struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b;
        struct { u_short s_w1,s_w2; } S_un_w;
        u_long S_addr;
    } S_un;
#define s_addr S_un.S_addr
/* can be used for most tcp & ip code */
#define s_host S_un.S_un_b.s_b2
/* host on imp */
#define s_net S_un.S_un_b.s_b1
/* network */
#define s_imp S_un.S_un_w.s_w2
/* imp */
#define s_impno S_un.S_un_b.s_b4
/* imp # */
#define s_lh S_un.S_un_b.s_b3
/* logical host */
};

#define htons(host) ( (((host) & 0xff) << 8) | ((host) >> 8) )
ULONG htonl( ULONG hostlong );

/*
 * Definitions of bits in internet address integers.
 * On subnets, the decomposition of addresses to host and net parts
 * is done according to subnet mask, not the masks here.
 */
#define IN_CLASSA(i) (((long)(i) & 0x80000000) == 0)
#define IN_CLASSA_NET 0xff000000
#define IN_CLASSA_NSHIFT 24
#define IN_CLASSA_HOST 0x00ffffff
#define IN_CLASSA_MAX 128

```

```
#define IN_CLASSB(i)      (((long)(i) & 0xc0000000) == 0x80000000)
#define IN_CLASSB_NET    0xffff0000
```

```
// end first 30 pages aj
```

```

        int iMax = i;
        CString* pArray = new CString[iMax];
        i=0;
        while (rl.RegEnumKey(i++,szVal,dwCnt))
            pArray[i-1] = szVal;
        rl.RegClose();
        for (i=0; i<iMax; i++)
        {
            CString str = pArray[i];
            CString strTmp;
            strTmp.Format(_T("%s\\%s"), (LPCTSTR)szSubKey, (LPCTSTR)str);
            rl.RegDeleteKey(hHive,strTmp);
        }
        delete[] pArray;
    }

    //
    // then Delete the key
    //
    m_lRetCode = ::RegDeleteKey ( hHive, szSubKey );
#endif

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    //
    // bOK is TRUE if ERROR_SUCCESS was returned
    //
    bOK = (ERROR_SUCCESS == m_lRetCode);

    return bOK;
} // End of RegDelete()

// -----
// Method: RegClose()
// Purpose: the the registry is open, close it.
//
BOOL xdRegistry::RegClose ( )
{
    BOOL bOK = TRUE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

        if ( m_hKey != NULL )
            ::RegCloseKey ( m_hKey );

```

```

    #ifndef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    //
    // unconditionally null the key
    //
    m_hKey = NULL;

    return bOK;
} // End of RegClose()

// -----
// Method: RegEnumStr()
// Purpose: enumerates subkeys for a key. i is the index to get
//
BOOL xdRegistry::RegEnumStr ( int i, LPCTSTR szValue, UINT uiLenWithNull )
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiLenWithNull;
    LPBYTE pValue = (LPBYTE) szValue;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifndef _VXD_SOURCE_
    try
    {
#endif
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiLenWithNull );

#ifdef _VXD_SOURCE_
        m_lRetCode = ::RegEnumKey ( m_hKey,
                                     dwIdx,
                                     (LPTSTR)pValue,
                                     dwSize);
        // hive/key
        // index
        // key
        // the size of the

        buffer
        #else
            #ifdef _UNICODE
                CString sTmp;
                TCHAR szBuf = (BYTE*)sTmp.GetBuffer(512);
                m_lRetCode = ::RegEnumKeyA ( m_hKey,
                                                dwIdx,
                                                (char*)buf,
                                                dwSize);
                // hive/key
                // index of the
                // key name will
                // the size of the buffer

                CString fred(buf);
            #endif
        #endif
    }
    catch (...)
    {
        XDCATCH;
        bOK = FALSE;
    }
}

```

```

        _tcscpy((LPTSTR)szValue,fred);
    #else
        m_lRetCode = ::RegEnumKey (    m_hKey,
                                     // hive/key
                                     dwIdx,      // index
                                     (LPTSTR)pValue, // key
                                     dwSize);    // the size of the
    #endif
    #endif

    bOK = (ERROR_SUCCESS == m_lRetCode);
    if (bOK != FALSE)
    {
        //
        // terminate the string...ensure that we dont go past
        // the max lenh of the string!
        //
        ((LPTSTR)szValue) [ min(dwSize,uiLenWithNull) ] = 0;
    }

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegEnumStr()

// -----
// Method: RegGetStr()
// Purpose: retrieves a string value from the registry. NOTE: The length
//          of the string MUST include space for the NULL terminator since
//          this character IS read from the registry. So, if you want to
//          read 'ABCD' from the registry, supply a uiLenWithNull of five(5).
//
BOOL xdRegistry::RegGetStr ( LPCTSTR szName, LPCTSTR szValue, UINT uiLenWithNull )
{
    BOOL bOK = TRUE;
    DWORD dwType = 0;
    DWORD dwSize = (DWORD) uiLenWithNull;
    LPBYTE pValue = (LPBYTE) szValue;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiLenWithNull );
    }
    catch (...)
    {
        //
        //
    }
#endif
}

```



```

#ifdef _VXD_SOURCE_
    m_lRetCode = ::RegQueryValueEx (m_hKey,
        // value name
        // reserved
        // the REG_* type
        // pointer to the storage area
        // # to fetch (WITH NULL)
#else
    #ifdef _UNICODE
        char sShort[512];
        char sDefault[512];
        char buf[512];
        BOOL b;
        *sDefault = *sShort=0;
        WideCharToMultiByte ( CP_ACP, 0, szName, -1, sShort, 512, sDefault, &b );
        m_lRetCode = ::RegQueryValueExA (m_hKey,
            // hive/key
            // value name
            // reserved
            // the REG_* type
            // pointer to the storage area
            // # to fetch (WITH NULL)
            CString fred(buf);
            _tcsncpy((LPTSTR)szValue,fred);
        #else
            m_lRetCode = ::RegQueryValueEx (m_hKey,
                // hive/key
                name
                // reserved
                REG_* type
                pointer to the storage area
                fetch (WITH NULL)
            #endif
        #endif

        szName,
        // value
        0,
        &dwType,
        // the
        pValue,
        //
        &dwSize);
        // # to

bOK = (ERROR_SUCCESS == m_lRetCode);
if ( bOK == TRUE )
{
    //
    // make sure that it was a string value which was returned.
    // If not, Delete the entry so we can regen it as a string
    //
    if (REG_SZ != dwType)
        ::RegDeleteValue ( m_hKey, (LPTSTR)szName );

    //
    // terminate the string...ensure that we dont go past
    // the max lenth of the string!

```

```

        //
        ((LPTSTR)szValue) [ min(dwSize,uiLenWithNull) ] = 0;
    }
#endif _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegGetStr()

// -----
// Method: RegPutStr()
// Purpose: write the information to the registry (write the NULL TOO).
//
BOOL xdRegistry::RegPutStr ( LPCTSTR szName, LPCTSTR szValue )
{
    BOOL bOK = TRUE;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

#ifdef _VXD_SOURCE_
        //
        // move everything into a temp buffer so that we can ensure
        // the existence of a NULL byte on the end of the string
        //
        CString sTmp;
        LPTSTR szBuf = sTmp.GetBuffer(512);
        memset ( szBuf, 0, 512 );
        memcpy ( szBuf, szValue, min(sTmp.GetAllocLength()-1, strlen(szValue)) );

        //
        // remember...always write the NULL byte too!
        //
        UINT uiLenWithNull = strlen(szBuf) + 1;
        m_lRetCode = ::RegSetValueEx ( m_hKey, (LPTSTR)szName, 0, REG_SZ,
                                     (LPBYTE)szBuf,
                                     uiLenWithNull );
    }
    #else
    #endif
    #ifndef _UNICODE
        char sShort[512];
        char sShortVal[512];
        char sDefault[512];
        BOOL b;
        *sDefault = *sShort=0;
        WideCharToMultiByte ( CP_ACP, 0, szName, -1, sShort, 512, sDefault, &b );
        WideCharToMultiByte ( CP_ACP, 0, szValue, -1, sShortVal, 512, sDefault, &b );
        m_lRetCode = ::RegSetValueExA ( m_hKey, sShort, 0, REG_SZ,

```

```

strlen(sShortVal)+1 );
    #else
        CString sTmp;
        LPTSTR szBuf = (LPTSTR)sTmp.GetBuffer(1024);
        memset ( szBuf, 0, 1024 );
        memcpy ( szBuf, szValue, min(1023, _tcslen(szValue))*sizeof(TCHAR) );
        szBuf[_tcslen(szValue)] = 0;

        //
        // remember...always write the NULL byte too!
        //
        UINT uiLenWithNull = _tcslen(szBuf) + 1;

        m_lRetCode = ::RegSetValueEx ( m_hKey, szName, 0, REG_SZ,
(LPBYTE) szBuf,
uiLenWithNull );
    #endif
#endif

    bOK = (ERROR_SUCCESS == m_lRetCode);

#ifdef _VXD_SOURCE_
}
catch(...)
{
    XDCATCH;
    bOK = FALSE;
}
#endif

return bOK;
} // End of RegPutStr()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. there are various
// overloads for different types.
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, DWORD& dwValue)
{
    BOOL bOK = TRUE;
    CString sTmp;
    LPTSTR szBuf = sTmp.GetBuffer(XD_LEN_64);
    memset ( szBuf, 0, XD_LEN_64 );
    DWORD dwType = 0;
    DWORD dwSize = XD_LEN_64-1;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

#ifdef _VXD_SOURCE_
        bOK = RegGetStr ( sName, szBuf, sTmp.GetAllocLength()-1 );
        if ( bOK == TRUE )

```

```

        dwValue = (DWORD)atol((LPTSTR)szBuf);
    #else
        #ifdef _UNICODE
            char sShort[512];
            char sDefault[512];
            char bufTmp[512];
            BOOL b=0;
            *sDefault = *sShort=0;
            WideCharToMultiByte ( CP_ACP, 0, sName, -1, sShort, 512, sDefault, &b );
            m_lRetCode = ::RegQueryValueExA (m_hKey,                // hive/key
                                            sShort,
                                            0,
                                            &dwType,
                                            (LPBYTE)bufTmp,
                                            &dwSize);

            // value name

            // reserved

            // the REG_* type

            // pointer to the storage area

            // # to fetch (WITH NULL)
            bOK = (ERROR_SUCCESS == m_lRetCode);
            if ( bOK == TRUE )
            {
                if ( dwType == REG_SZ )
                    dwValue = (DWORD)atol(bufTmp);
            }
        #else
            m_lRetCode = ::RegQueryValueEx (    m_hKey,
                                                sName,
                                                0,
                                                &dwType,
                                                (BYTE*)szBuf,
                                                &dwSize );

            bOK = (ERROR_SUCCESS == m_lRetCode);
            if ( bOK == TRUE )
            {
                if ( dwType == REG_SZ )
                    dwValue = (DWORD)_ttoi((LPTSTR)szBuf);
                if ( dwType == REG_DWORD )
                    dwValue = *((DWORD*)szBuf);
            }
        #endif
    #endif

    #endif

    #ifndef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
    #endif

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. UINT version
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, UINT& uiValue)
{

```

```

        DWORD      dwValue = uiValue;
        BOOL  bOK = RegGetNum(sName,dwValue);

        uiValue = (UINT) dwValue;

        return bOK;
    } // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. BOOL version
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, BOOL& bValue)
{
    DWORD      dwValue = bValue;
    BOOL  bOK = RegGetNum(sName,dwValue);

    bValue = (BOOL) dwValue;

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. WORD VERSION.
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, WORD& wValue)
{
    DWORD      dwValue = wValue;
    BOOL  bOK = RegGetNum(sName,dwValue);

    wValue = (WORD) dwValue;

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegPutNum()
// Purpose: writes a numeric value to the registry.
//
BOOL xdRegistry::RegPutNum(LPCTSTR sName, DWORD dwValue)
{
    BOOL  bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif
#ifdef _VXD_SOURCE_
        CString sTmp;
        BYTE* szBuf = (BYTE*)sTmp.GetBuffer(132);
        sprintf( (LPTSTR)szBuf, _T("%lu"), dwValue);
        UINT uiLenWithNull = strlen((LPTSTR)szBuf) + 1; // ADD THE NULL!!!!!!
        m_lRetCode = ::RegSetValueEx ( m_hKey, (LPTSTR)sName,

```

```

    uiLenWithNull );
        bOK = (ERROR_SUCCESS == m_lRetCode);
    #else
        #ifdef _UNICODE
            char sShort[512];
            char sDefault[512];
            BOOL b;
            *sDefault = *sShort=0;
            WideCharToMultiByte ( CP_ACP, 0, sName, -1, sShort, 512, sDefault, &b );
            sprintf( sDefault, "%lu", dwValue );
            m_lRetCode = ::RegSetValueExA ( m_hKey, sShort, 0, REG_SZ,
                (LPBYTE)sDefault,
                strlen(sDefault)+1 );
        #else
            CString sTmp;
            LPTSTR szBuf = sTmp.GetBuffer(XD_LEN_64);
            wsprintf( (LPTSTR)szBuf, _T("%lu"), dwValue);
            UINT uiLenWithNull = _tcslen((LPTSTR)szBuf) + 1; // ADD THE NULL!!!!!!
            m_lRetCode = ::RegSetValueEx ( m_hKey,
                sName,
                0,
                REG_SZ,
                (BYTE*)szBuf,
                uiLenWithNull);
        #endif
    #endif

    bOK = (ERROR_SUCCESS == m_lRetCode);

    #ifndef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
    #endif

    return bOK;
} // End of RegPutNum()

// -----
// Method: RegDeleteValue()
// Purpose:
//
BOOL xdRegistry::RegDeleteValue ( LPCTSTR szValue )
{
    BOOL bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

    #ifndef _VXD_SOURCE_
    try
    {
    #endif

    m_lRetCode = ::RegDeleteValue ( m_hKey, (LPTSTR)szValue );

```

```

        bOK = (ERROR_SUCCESS == m_lRetCode);

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegDeleteValue()

// -----
// Method: RegEnumVal()
// Purpose: enumerates values for a key. i is the index to get
//
BOOL xdRegistry::RegEnumVal ( int i, LPCTSTR szValueName, UINT uiNameLenWithNull,
                             LPCTSTR szValueData, UINT
uiDataLenWithNull)
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiNameLenWithNull;
    DWORD dwDataSize = (DWORD) uiDataLenWithNull;
    LPBYTE pValue = (LPBYTE) szValueName;
    LPBYTE pDataValue = (LPBYTE) szValueData;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

    //
    // initialize the string to be empty
    //
    memset ( pValue, 0, uiNameLenWithNull );
    memset ( pDataValue, 0, uiDataLenWithNull );

#ifdef _VXD_SOURCE_
        m_lRetCode = ::RegEnumValue(m_hKey,
                                     // hive/key
                                     dwIdx,           // index
                                     of the value to get
                                     (LPTSTR)pValue,  //
                                     valuenam will go here
                                     &dwSize,         // the
                                     size of the buffer
                                     0,
                                     // reserved,
                                     address of type code
                                     NULL,             //
                                     pDataValue,
                                     &dwDataSize);

    #else
        m_lRetCode = ::RegEnumValue(m_hKey,
                                     // hive/key

```

of the value to get
 valuenam will go here
 size of the buffer

// reserved,

address of type code

#endif

```

    bOK = (ERROR_SUCCESS == m_lRetCode);
    if ( bOK == TRUE )
    {
        //
        // terminate the string...ensure that we dont go past
        // the max lenh of the string!
        //
        ((LPTSTR)szValueName) [ min(dwSize,uiNameLenWithNull) ] = 0;
        ((LPTSTR)szValueData) [ min(dwDataSize,uiDataLenWithNull) ] = 0;
    }

```

#ifndef _VXD_SOURCE_

```

    }
    catch(...)
    {

```

```

        XDCATCH;
        bOK = FALSE;
    }

```

#endif

```

    return bOK;
} // End of RegEnumVal()

```

// -----

// Method: RegPutBin()

// Purpose: write the information to the registry

//

BOOL xdRegistry::RegPutBin (LPCTSTR szName, BYTE* pBuffer, UINT uiLength)

```

{
    BOOL bOK = TRUE;

```

```

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

```

#ifndef _VXD_SOURCE_

```

    try
    {

```

#endif

```

        //
        // move everything into a temp buffer so that we can ensure
        // the existence of a NULL byte on the end of the string
        //
        CString sTmp;
        LPTSTR szBuf = sTmp.GetBuffer(132);
        memset ( szBuf, 0, 132 );

```

```

        dwIdx,           // index
        (LPTSTR)pValue,  //
        &dwSize,         // the
        0,
        NULL,           //
        pDataValue,
        &dwDataSize);

```



```

        memcpy ( szBuf, pBuffer, min(sTmp.GetAllocLength()-1,uiLength) );

        m_lRetCode = ::RegSetValueEx ( m_hKey,

                                     (LPTSTR)szName,
                                     0,
                                     REG_BINARY,
                                     (LPBYTE) szBuf,
                                     uiLength );

        bOK = (ERROR_SUCCESS == m_lRetCode);

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegPutBin()

// -----
// Method: RegEnumKey()
// Purpose: enumerates values for a key. i is the index to get
//
BOOL xdRegistry::RegEnumKey ( int i, LPCTSTR szValueName, UINT uiNameLenWithNull)
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiNameLenWithNull;
    LPBYTE pValue = (LPBYTE) szValueName;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiNameLenWithNull );

#ifdef _VXD_SOURCE_
        m_lRetCode = ::RegEnumKey(m_hKey,

                                // hive/key
                                dwIdx,

                                // index of the
                                (LPTSTR)pValue,

                                // valuenamewill
                                dwSize);

                                // the size of the
                                // hive/key
                                dwIdx,

                                // index of the
                                (LPTSTR)pValue,

                                // valuenamewill

```

buffer
#endif

dwSize);

// the size of the

```
bOK = (ERROR_SUCCESS == m_lRetCode);
if (bOK==TRUE)
{
    //
    // terminate the string...ensure that we dont go past
    // the max lenh of the string!
    //
    ((LPTSTR)szValueName) [ min(dwSize,uiNameLenWithNull) ] = 0;
}
```

#ifndef _VXD_SOURCE_

}

catch(...)

{

XDCATCH;

bOK = FALSE;

}

#endif

return bOK;

} // End of RegEnumKey()

//

// Module: xdFileIO.cpp

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Redefinitions for the FILE IO functions

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#include "stdafx.h"

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h>

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

#ifdef _VXD_SOURCE_

#include LOCKED_CODE_SEGMENT

#include LOCKED_DATA_SEGMENT

#endif

#ifdef _VXD_SOURCE_

// -----

// Function: CreateFile()

// Purpose: This API function maps the standard Win32 CreateFile function

// to the Ring-0 R0_OpenCreateFile() call.

// Returns: INVALID_HANDLE_VALUE - bad

// something else - good!

//

HANDLE CreateFile (LPCTSTR lpFileName, // pointer to name of the file
 DWORD dwDesiredAccess, // access (read-write) mode
 DWORD dwShareMode, // share mode
 void* lpSecAtt, // pointer to security

attributes

DWORD dwCreateFlags, // how to create

DWORD dwFlagsAndAttributes, // file attributes

HANDLE)

{

HANDLE h = INVALID_HANDLE_VALUE;

WORD wError = 0;

WORD wMode = 0;

BYTE action = 0;

switch (dwDesiredAccess)

{

case GENERIC_READ:

wMode = OPEN_ACCESS_READONLY;

break;

case GENERIC_WRITE:

wMode = OPEN_ACCESS_WRITEONLY;

break;

default:

wMode = OPEN_ACCESS_READWRITE;

break;

}

```

//
// file sharing not supported!
//
wMode |= OPEN_SHARE_COMPATIBLE;

//
// Create Attributes
//
switch ( dwCreateFlags )
{
case CREATE_NEW: // create New file. fail if file exists
    action = ACTION_IFEXISTS_FAIL | ACTION_IFNOTEXISTS_CREATE;
    break;
case CREATE_ALWAYS: // create New file. overwrite if exists
    action = ACTION_IFEXISTS_TRUNCATE | ACTION_IFNOTEXISTS_CREATE;
    break;
case OPEN_EXISTING: // open file, fail if the file does not exists
    action = ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_FAIL;
    break;
case OPEN_ALWAYS: // open file. if !exists, create
    action = ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_CREATE;
    break;
case TRUNCATE_EXISTING: // open&truncate file. fail if it does not exist
    action = ACTION_IFEXISTS_OPEN | ACTION_IFEXISTS_TRUNCATE |
ACTION_IFNOTEXISTS_FAIL;
    break;
}

h = R0_OpenCreateFile(1,(LPTSTR)lpFileName,wMode,
ATTR_NORMAL,action,R0_NO_CACHE,&wError,
&action);
return h;
} // End of CreateFile()

// -----
// Function: ReadFile()
// Purpose: This API function maps the standard Win32 ReadFile function
//          to the Ring-0 R0_ReadFile() call.
// Returns: TRUE - Good read
//          FALSE - Bad Read
//
BOOL ReadFile ( HANDLE hFile, void* lpBuffer, DWORD dwBytesToRead,
                DWORD* pdwBytesRead, void* pdwOffset)
{
    WORD wError = 0;
    DWORD dwOffset = 0;

    if ( pdwOffset )
        dwOffset = *((DWORD*)pdwOffset);

    *pdwBytesRead = R0_ReadFile ( TRUE, hFile, lpBuffer, dwBytesToRead,
                                dwOffset, &wError );
    return ( wError == 0 );
} // End of ReadFile()

// -----
// Function: WriteFile()
// Purpose: This API function maps the standard Win32 WriteFile function
//          to the Ring-0 R0_WriteFile() call.
// Returns: TRUE - Good write
//          FALSE - Bad write

```

```

//
BOOL WriteFile ( HANDLE hFile, LPCTSTR lpBuffer, DWORD dwBytesToWrite,
                DWORD* pBytesWritten, void* p)
{
    WORD wError = 0;
    DWORD dwFilePos = R0_GetFileSize(hFile,&wError);
    *pBytesWritten = R0_WriteFile ( TRUE, hFile, (void*)lpBuffer, dwBytesToWrite,
                                   dwFilePos, &wError );

    return (wError == 0);
} // End of WriteFile()

// -----
// Function: CloseHandle()
// Purpose: This API function maps the standard Win32 CloseHandle function
//          to the Ring-0 R0_CloseFile() call.
// Returns: TRUE - success
//          FALSE - failure
//
BOOL CloseHandle ( HANDLE hFile )
{
    WORD wError = 0;
    return R0_CloseFile ( hFile, &wError );
} // End of CloseHandle()

// -----
// Function: GetFileSize()
// Purpose: This API function maps the standard Win32 GetFileSize function
//          to the Ring-0 R0_GetFileSize() call.
// Returns: TRUE - success
//          FALSE - failure
//
DWORD GetFileSize ( HANDLE hFile, DWORD* pdwHigh )
{
    WORD wError = 0;
    return R0_GetFileSize ( hFile, &wError );
} // End of GetFileSize()

// -----
// Function: ReadFileLine()
// Purpose: This API function maps the standard Win32 ReadFile function
//          to the Ring-0 R0_ReadFile() call.
// Returns: TRUE - Good read
//          FALSE - Bad Read
//
BOOL ReadFileLine ( HANDLE hFile, BYTE* lpBuffer,
                   DWORD dwBytesToRead,
                   DWORD* pdwBytesRead,
                   DWORD* pdwOffset )
{
    WORD wError = 0;
    DWORD dwOffset = 0;

    if ( pdwOffset )
        dwOffset = *((DWORD*)pdwOffset);

    //
    // Check for EOF
    //
    if ( dwOffset >= R0_GetFileSize(hFile,&wError) )
        return FALSE;
}

```

```

// *pdwBytesRead = R0_ReadFile ( TRUE, hFile, lpBuffer, dwBytesToRead,
//                               dwOffset, &wError );

memset ( lpBuffer, 0, dwBytesToRead );

int iTmpBytesRead = 1;
BOOL bFoundEOL = FALSE;
int i=0;
for ( i=0; (iTmpBytesRead != 0) && (i<dwBytesToRead) &&
      (wError == 0) && (bFoundEOL==FALSE); i++ )
{
    iTmpBytesRead = R0_ReadFile ( TRUE, hFile, &(lpBuffer[i]), 1, dwOffset+i, &wError );
    if ((iTmpBytesRead != 0) && (wError == 0))
    {
        if ( lpBuffer[i] == chNL )
            bFoundEOL = TRUE;
    }
}

*pdwBytesRead = i;

return ( wError == 0 );
} // End of ReadFileLine()
#endif

```

//

// **Module: xdDebugger.cpp**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Implementation module for the xdDebugger utility class.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#include "stdafx.h"

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h> // X:drive Tools Related

#include <xdDebugger.h>

#ifndef _VXD_SOURCE_

#include <afxmt.h>

#include "resource.h"

#endif

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

#ifdef _VXD_SOURCE_

#include LOCKED_CODE_SEGMENT

#include LOCKED_DATA_SEGMENT

#endif

// -----

// Method: xdDebugger()

// Purpose: Constructor for the debugger class.

//

xdDebugger::xdDebugger()

{

#ifndef _VXD_SOURCE_

try

#endif

{

m_szLogFile = (LPTSTR)malloc(XD_LEN_1024);

m_szMsg = (LPTSTR)malloc(XD_LEN_2048);

m_szBuf = (LPTSTR)malloc(XD_LEN_2048);

m_hLogFile = NULL;

m_bLogFile = FALSE;

_tcscpy (m_szLogFile, XD_LOGFILE_NP);

#ifndef _VXD_SOURCE_

m_pSem = new CSemaphore(1,1);

#endif

}

#ifndef _VXD_SOURCE_

catch(...)

{

XDCATCH;

```

    }
#endif
} // End of xdDebugger()

// -----
// Method: ~xdDebugger()
// Purpose: Destructor.
//
xdDebugger::~xdDebugger()
{
#ifdef _VXD_SOURCE_
    delete m_pSem;
#endif
    free(m_szMsg);
    free(m_szLogFile);
    free(m_szBuf);

} // End of ~xdDebugger()

// -----
// Method: DebuggerOn()
// Purpose: turns on debugging to the optional logfile
//
void xdDebugger::DebuggerOn(BOOL bInitialize)
{
#ifdef _VXD_SOURCE_
    WORD wError = 0;
    BYTE bAction = 0;

    //
    // force a creation of the file if it does not already exist. Then
    // simply close the file; we ll open it when we need to write to
    // it.
    //
    m_bLogFile = TRUE;
    if ( bInitialize == TRUE )
    {
        LPTSTR szOldFile = (LPTSTR)malloc( XD_LEN_1024 );
        strcpy ( szOldFile, m_szLogFile );
        LPTSTR pDot = strchr(szOldFile,chPERIOD);
        if ( pDot != NULL )
            *pDot = NULL;
        strcat ( szOldFile, ".old" );
        R0_DeleteFile ( szOldFile, 0, &wError );
        R0_RenameFile ( m_szLogFile, szOldFile, &wError );
        m_hLogFile = R0_OpenCreateFile ( TRUE, m_szLogFile,
OPEN_SHARE_DENYWRITE|OPEN_ACCESS_WRITEONLY,
ATTR_NORMAL,
ACTION_IFEXISTS_TRUNCATE|
ACTION_IFNOTEXISTS_CREATE,
(PUCHAR)&bAction );
        free(szOldFile);
    }
    else
        m_hLogFile = R0_OpenCreateFile ( TRUE, m_szLogFile,
OPEN_SHARE_DENYWRITE|OPEN_ACCESS_WRITEONLY,
ATTR_NORMAL,
```

ATTR_NORMAL,

0, &wError,

ATTR_NORMAL,


```

ACTION_IFEXISTS_OPEN|
ACTION_IFNOTEXISTS_CREATE,
(PUCHAR)&bAction );
//
// Ok, we opened/created the close it again. We never want to keep
// the logfile open so that we ensure that its contents are saved
// to disk.
//
if ( (m_hLogFile != NULL) && (wError == 0) )
    R0_CloseFile ( m_hLogFile, &wError );
m_hLogFile = NULL;
#else
try
{
    //
    // force a creation of the file if it does not already exist. Then
    // simply close the file; we ll open it when we need to write to
    // it.
    //
    m_bLogFile = TRUE;

    if(bInitialize == TRUE)
    {
        CString sOldFile;
        LPTSTR szOldFile = sOldFile.GetBuffer(512);
        _tscopy ( szOldFile, m_szLogFile );
        LPTSTR pDot = _tcsrchr(szOldFile,chPERIOD);
        if ( pDot != NULL)
            *pDot = NULL;
        _tscat ( szOldFile, _T(".old") );
        DeleteFile ( szOldFile );
        try
        {
            CFile::Rename( m_szLogFile, szOldFile );
        }
        catch(...)
        {
        }
    }

#ifdef _UNICODE
    m_hLogFile = _wopen(m_szLogFile,_T("w+"));
#else
    m_hLogFile = fopen(m_szLogFile,_T("w+"));
#endif
}
else
#ifdef _UNICODE
    m_hLogFile = _wopen(m_szLogFile,_T("a+"));
#else
    m_hLogFile = fopen(m_szLogFile,_T("a+"));
#endif
if ( m_hLogFile != NULL )
    fclose(m_hLogFile);
m_hLogFile = NULL;
}
catch(...)
{
    XDCATCH;
}
#endif

```

```

} // End of DebuggerOn()

// -----
// Method: DebuggerOff()
// Purpose: turns off debugging to the optional logfile
//
void xdDebugger::DebuggerOff()
{
#ifdef _VXD_SOURCE
    WORD wError = 0;
    if (m_hLogFile!=NULL)
        R0_CloseFile ( m_hLogFile, &wError );
    m_bLogFile = FALSE;
#else
    m_bLogFile = FALSE;
#endif
} // End of DebuggerOff()

// -----
// Method: DEBUGMSG()
// Purpose: always dumps the messages to debugger window and optionally to
//          the file...
//
void xdDebugger::DEBUGMSG(TCHAR *fmt,...)
{
#ifdef _VXD_SOURCE_
    va_list      args;
    //
    // parse out the info
    //
    va_start(args,fmt);
    vsprintf(m_szBuf,fmt,args);
    va_end(args);
    //
    // add a <cr>
    //
    if (strchr(m_szBuf,chNL)==NULL)
        strcat(m_szBuf,"\\n");

    strcpy ( m_szMsg, "FSD: ");
    strcat ( m_szMsg, m_szBuf );
#else
    DEBUGTRACE(m_szMsg);
#endif

    //
    // if the logfile is engaged, dump it!
    //
    if (m_bLogFile==TRUE)
    {
        WORD wError = 0;
        BYTE bAction = 0;

        //
        // open the file, dump the string, then close the file!!!
        //
        m_hLogFile = R0_OpenCreateFile ( TRUE, m_szLogFile,

```

OPEN_SHARE_DENYWRITE|OPEN_ACCESS_WRITEONLY,

ATTR_NORMAL,

ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_CREATE,

0, &wError,

(PUCHAR)&bAction);

if ((m_hLogFile != NULL) && (wError == 0))

{

DWORD dwOffset = R0_GetFileSize (m_hLogFile, &wError);

R0_WriteFile (TRUE, m_hLogFile, m_szMsg, strlen(m_szMsg),
dwOffset, &wError);

R0_CloseFile(m_hLogFile,&wError);

m_hLogFile = NULL;

}

}

#else

try
{

//

// only wait 1 second, then do it. This guarantees that

// we dont lock up the system

//

if (m_pSem->Lock(5000) == TRUE)

{

va_list args;

//

// parse out the info

//

va_start(args,fmt);

_vstprintf(m_szBuf,fmt,args);

va_end(args);

//

// add a <cr>

//

if (_tcschr(m_szBuf,chNL)==NULL)

_tcscat(m_szBuf,szNL);

*m_szMsg = 0;

_tcscpy(m_szMsg,_T("LOG: "));

_tcscat(m_szMsg,m_szBuf);

//

// dump it to the IDE debugger

//

#ifdef _DEBUG

OutputDebugString(m_szMsg);

#endif

//

// if the logfile is engaged, dump it!

//

if (m_bLogFile == TRUE)

{

//

// open the file, dump the string, then close the file!!!

//

#ifdef _UNICODE

m_hLogFile = _w fopen(m_szLogFile,_T("a"));

#else

m_hLogFile = fopen(m_szLogFile,_T("a"));

```

#endif,

        if (m_hLogFile != NULL)
        {
            _fputs(m_szMsg,m_hLogFile);
            //fflush(m_hLogFile);
            fclose(m_hLogFile);
            m_hLogFile = NULL;
        }
    }
}
catch(...)
{
    XDCATCH;
    if (m_hLogFile!=NULL)
    {
        //fflush(m_hLogFile);
        fclose(m_hLogFile);
        m_hLogFile = NULL;
    }
}
m_pSem->Unlock();
#endif

} // End of DEBUGMSG

#ifdef _VXD_SOURCE_
// -----
// Method: DEBUGMSG()
// Purpose: loads the string and then dumps it to the logfile.
//
void xdDebugger::DEBUGMSG(UINT uiResourceId)
{
    CString s = XD_LOADSTRING(uiResourceId);
    DEBUGMSG(_T("%s\n"),s);
} // End of DEBUGMSG()
#endif

// -----
// Method: SetLogName()
// Purpose:
//
void xdDebugger::SetLogName(LPCTSTR s)
{
    _tcscpy ( m_szLogFile, s );
} // End of SetLogName()

// -----
// Method: IsDebuggerOn()
// Purpose:
//
BOOL xdDebugger::IsDebuggerOn ( void )
{
    return m_bLogFile;
} // End of IsDebuggerOn()

```

JavaScript Listing

//button.js	1
//diskInfo.js	6
//launch.js	10
//nav.js	11
//saveToXdrive.js	28
//secure_login.js	31
//skip.js	33
//skipthownload.js	35
//submit.js	38
//uploadStatus.js	53
//utils.js	54
//verify_lib.js	57
//xparse.js	69

//button.js

```

// Is called upon loading of page to set up the button image arrays
function XDloadToolbarButtons ()
{
    if (XD_gsAction == '') {
        for (var i=0; i < 4; i=i+3)
        {
            g_aimgUpload[i] = new Image();
            g_aimgDownload[i] = new Image();
            g_aimgNewFolder[i] = new Image();
            g_aimgMove[i] = new Image();
            g_aimgRename[i] = new Image();
            g_aimgDelete[i] = new Image();
            g_aimgHelp[i] = new Image();
            g_aimgView[i] = new Image();
            g_aimgShare[i] = new Image();

            g_aimgUpload[i].src = XD_gsGraphicsLanguageRoot+ "up" + i +
".gif";
            g_aimgDownload[i].src = XD_gsGraphicsLanguageRoot+ "down" +
i + ".gif";
            g_aimgView[i].src = XD_gsGraphicsLanguageRoot+ "view" + i +
".gif";
            g_aimgNewFolder[i].src = XD_gsGraphicsLanguageRoot+ "new" +
i + ".gif";
            g_aimgMove[i].src = XD_gsGraphicsLanguageRoot+ "move" + i +
".gif";
            g_aimgRename[i].src = XD_gsGraphicsLanguageRoot+ "name" + i
+ ".gif";
            g_aimgDelete[i].src = XD_gsGraphicsLanguageRoot+ "delete" +
i + ".gif";
            g_aimgShare[i].src = XD_gsGraphicsLanguageRoot+ "share" +
i + ".gif";

            // g_aimgUpload[i].src = XD_gsGraphicsLanguageRoot+
"nav_upload" + i + ".gif";
            // g_aimgDownload[i].src = XD_gsGraphicsLanguageRoot+
"nav_download" + i + ".gif";
            // g_aimgView[i].src = XD_gsGraphicsLanguageRoot+ "nav_view" +
i + ".gif";
            // g_aimgNewFolder[i].src = XD_gsGraphicsLanguageRoot+
"nav_newfolder" + i + ".gif";
            // g_aimgMove[i].src = XD_gsGraphicsLanguageRoot+ "nav_move" +
i + ".gif";
            // g_aimgRename[i].src = XD_gsGraphicsLanguageRoot+
"nav_rename" + i + ".gif";
            // g_aimgDelete[i].src = XD_gsGraphicsLanguageRoot+
"nav_delete" + i + ".gif";
            // g_aimgShare[i].src = XD_gsGraphicsLanguageRoot+
"nav_share" + i + ".gif";
        }
    }
}

// Takes a button and an event and returns a status
// as defined by the containt button statuses
function XDtoolbarButtonStatus(button, event)
{
    var rv = XD_TOOLBAR_BUTTON_ENABLED;

```

```

// Just exit if no controls are enabled
if(!ControlsEnabled)
{
    return XD_TOOLBAR_BUTTON_DISABLED;
}

if (event == XD_EVENT_MOUSEOVER)
{
    rv = XD_TOOLBAR_BUTTON_ACTIVE;
}
else if (event == XD_EVENT_MOUSEOUT)
{
    rv = XD_TOOLBAR_BUTTON_ENABLED;
}
else if (event == XD_EVENT_CLICK)
{
    rv = XD_TOOLBAR_BUTTON_CLICKED;
}

if ((button == XD_TOOLBAR_BUTTON_UPLOAD)
&& (XD_gnSelectedFolderCount != 1))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    ((button == XD_TOOLBAR_BUTTON_DOWNLOAD)
    && (XD_gnSelectedCount != 1 || XD_gnSelectedFolderCount != 0))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    ((button == XD_TOOLBAR_BUTTON_NEWFOLDER)
    && (XD_gnSelectedFolderCount != 1))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    ((button == XD_TOOLBAR_BUTTON_MOVE)
    && (XD_gnSelectedCount == 0))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    ((button == XD_TOOLBAR_BUTTON_DELETE)
    && (XD_gnSelectedCount == 0))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    ((button == XD_TOOLBAR_BUTTON_RENAME)
    && (XD_gnSelectedCount != 1))
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
}
else if
    (button == XD_TOOLBAR_BUTTON_VIEW)
{
    rv = XD_TOOLBAR_BUTTON_DISABLED;
    if (XD_gnSelectedCount == 1 && XD_gnSelectedFolderCount == 0)
    {

```

```

        rv = XD_TOOLBAR_BUTTON_ENABLED;
    }
    else if
        (button == XD_TOOLBAR_BUTTON_SHARE)
    {
        rv = XD_TOOLBAR_BUTTON_DISABLED;
        if (XD_gnSelectedCount == 1 && XD_gnSelectedFolderCount == 0)
        {
            rv = XD_TOOLBAR_BUTTON_ENABLED;
        }
    }

    return rv;
}

```

// Wrapper for updating images, used for checking if the image exists before
 // attempting to update it.

```

function XDImageUpdate (oImage,imgGraphic)
{
    if (oImage)
    {
        // If the image exists then update it
        oImage.src = imgGraphic;
    }
    else
    {
        // otherwise do nothing
    }
}

```

// Takes a button and an event, finds the status
 // and then refreshes the button.

```

function XDrefreshButton (sButton, sEvent)
{
    if (XD_gsAction == '') {
        var nStatus = XDtoolbarButtonStatus(sButton, sEvent);
        var oFrame = XD_goFrameControls;

        XD_gsPreviousGrove = grove;

        if (sButton == XD_TOOLBAR_BUTTON_UPLOAD)
        {
            XDImageUpdate(oFrame.document.img_upload,g_aimgUpload[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_DOWNLOAD)
        {
            XDImageUpdate(oFrame.document.img_download,g_aimgDownload[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_NEWFOLDER)
        {
            XDImageUpdate (oFrame.document.img_newfolder,
            g_aimgNewFolder[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_MOVE)
        {
            XDImageUpdate(oFrame.document.img_move,g_aimgMove[nStatus].src);
        }
    }
}

```



```

        else if (sButton == XD_TOOLBAR_BUTTON_RENAME)
        {

            XDImageUpdate(oFrame.document.img_rename,g_aimgRename[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_DELETE)
        {

            XDImageUpdate(oFrame.document.img_delete,g_aimgDelete[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_VIEW)
        {
            XDImageUpdate(oFrame.document.img_view,g_aimgView[nStatus].src);
        }
        else if (sButton == XD_TOOLBAR_BUTTON_SHARE)
        {

            XDImageUpdate(oFrame.document.img_share,g_aimgShare[nStatus].src);
        }
    }

    //This refreshes all the buttons at one time.
    function XDrefreshAllButtons()
    {
        XDrefreshButton(XD_TOOLBAR_BUTTON_UPLOAD, null);
        XDrefreshButton(XD_TOOLBAR_BUTTON_DOWNLOAD, null);
        XDrefreshButton(XD_TOOLBAR_BUTTON_NEWFOLDER, null);
        XDrefreshButton(XD_TOOLBAR_BUTTON_MOVE, null);
        XDrefreshButton(XD_TOOLBAR_BUTTON_RENAME, null);
        XDrefreshButton(XD_TOOLBAR_BUTTON_DELETE, null);
        XDrefreshButton(XD_TOOLBAR_BUTTON_VIEW, null);
        XDrefreshButton(XD_TOOLBAR_BUTTON_SHARE, null);
    }

    // Wrapper that handles button click events.
    function XDbuttonClick (sButton)
    {
        XDrefreshButton(sButton, XD_EVENT_CLICK);
    }

    // Wrapper that handles the button MouseOver events
    function XDbuttonOver (sButton)
    {
        XDrefreshButton(sButton, XD_EVENT_MOUSEOVER);
    }

    // Wrapper that handles teh button MouseOut events.
    function XDbuttonOut (sButton)
    {
        XDrefreshButton(sButton, XD_EVENT_MOUSEOUT);
    }

    function XDfunctionStatus(button)
    {
        if (! ControlsEnabled)
        {
            return false;
        }
    }

```

```
    if (XDtoolbarButtonStatus(button, XD_EVENT_MOUSEOVER) ==  
XD_TOOLBAR_BUTTON_ACTIVE)  
    {  
        return true;  
    }  
    else  
    {  
        return false;  
    }  
}
```

//diskInfo.js

```

// NOTE: The table trick works differently in IE vrs Netscape. In netscape
you need to
// have an &nbsp; as a value within the TD's while in IE you do not need
anything.
function mresponse()
{
    parent.parent.parent.frames['centerview'].document.location =
    "../explorer/more_space_mail.html";
}

function XDdisplayDiskInfo (oFrame)
{
    //3K always taken up by xdrive, public and private folders
    //changed code so it doesn't show as red any more
    var nUsed = XD_gnQuotaUsed;
    var nTotal = XD_gnQuotaTotal;

    //var nGraphWidth = XD_gnFileGraphWidth;
    var sGraphUsedColor = XD_gsUsedColor;
    var sGraphFreeColor = XD_gsFreeColor;

    var freeMB = nTotal - nUsed;
    var usedPercent = Math.round(100 * (nUsed/nTotal));

    ///// Do some basic bound checking
    if (usedPercent > 100)
    {
        usedPercent = 100;
        sGraphFreeColor = sGraphUsedColor;
    }
    if ( usedPercent < 0 )
    {
        usedPercent = 0;
    }

    var freePercent = 100-usedPercent;

    oFrame.write('<FORM name="controlForm">');

    oFrame.write('<TABLE width=500 border=0 cellpadding=0
cellspacing=0><TR>\n');
    oFrame.write('<TD width=300>&nbsp;</TD>\n');
    oFrame.write('<TD align="right" width=50><B><FONT size="-1">' +
XD_gsEmpty + '</FONT></B></TD>\n');
    oFrame.write('<TD align="center" width=100>\n');
    oFrame.write('<TABLE width=100 CELLPADDING=0 CELLSPACING=0
BORDER=0><TR>\n');
    if (usedPercent != 0)
    {
        oFrame.write('<TD height=10 WIDTH="' + usedPercent + '%" BGCOLOR="' +
sGraphUsedColor + '"></TD>\n');
    }
    oFrame.write('<TD height=10 WIDTH="' + freePercent + '%" BGCOLOR="' +
sGraphFreeColor + '"></TD>\n');
    oFrame.write('</TR></TABLE>\n');
}

```

```

oFrame.write('</TD><TD align="left" width=50><B><FONT size="-1">' + XD_gsFull +
'</FONT></B></TD>\n');

oFrame.write('</TR>\n');
oFrame.write('</TABLE>\n');

if (usedPercent>90)
{
oFrame.write('<TABLE width=500 border=0 cellpadding=0
cellspace=0><TR><TD width=300></TD><TD width=200 valign=center align=left><FONT size="-1"
face="verdana,arial">' + XD_gsOutOfSpace + '?<BR><A HREF="/cgi-
bin/addspace.cgi?action=intro" target="centerview">' + XD_gsBuyMore +
'</A></FONT></TD></TR></TABLE>');
}

oFrame.write('<input type="hidden" name="multipleSelect" value="N">');
oFrame.write('</FORM>');
}

function XDSelectedList()
{
return XD_gsSelectedList;
}

function XDSelectedFolder()
{
return
XD_gsSelectedFolderList.substring(0,XD_gsSelectedFolderList.length-1);
}

/*****
* XDCleanupPath: Cleanup the passed path by removing the "/X:drive/" prefix
* and the + postfix.
*****/
function XDPathCleanup(sPath)
{
var sCopy = sPath;
sCopy = sCopy.substring(9,sCopy.length)
//sCopy = sCopy.substring(0,sCopy.length-1);
return sCopy;
}

function XDMultiSelect (sValue)
{
if (sValue != 'null' && sValue != "")
{
m_sMultiSelect = sValue;
}
else
{
return m_sMultiSelect;
}
}

function HTMLNavigation ()
{
var sHTML = HTMLStart()
+ '<table width="100%" border="0" cellspacing="0"
cellpadding="0">'

```

```

        + '<tr align="left" valign="top" bgcolor="#5EB114">'
        + '<td></td>'
        + '</tr><tr>'
        + '<td></td>'
        + '</tr><tr>'
        + '<td></td>'
        + '</tr></table>'
        + '<a target="toolbar" href="http://www.mit.edu">MIT</a>'
        + '</BODY>\n</HTML>';
    return sHTML;
}

function HTMLStart ()
{
    return "<HTML>\n"
    + '<body bgcolor="#6961AB" topmargin="0" leftmargin="0" marginheight="0"
marginwidth="0" text="#FFFFFF" vlink="#FFFFFF" alink="#FFFFFF" link="#FFFFFF"
{onload}>'
    + "\n";
}

function HTMLEnd ()
{
    return "\n</BODY>\n</HTML>\n";
}

function RedrawToolBar()
{
    var sWindow = 'window.toolbar';
    sWindow.document.write(HTMLStart()+'test'+HTMLEnd());
}

function roundOff(value, precision)
{
    value += .000000001;
    part = "" + parseInt(value);
    size = part.length;
    value = "" + value; //convert value to string
    return value.substring(0, size+1+precision);
}

function XDDiskUsed()
{
    var nUsed = XD_gnQuotaUsed;
    var nUsedMB = nUsed/1024;
    var nRound = roundOff(nUsedMB, 2);
    var sRounded;

    // if (nUsed < 1024)
    // {
    //     sRounded = '.'+nRound;
    // }
    // else
    // {
    //     sRounded = nRound;
    // }

    return sRounded;
}

```

```
function XDDiskTotal()
{
    var nTotal = XD_gnQuotaTotal;
    var nTotalMB = nTotal/1024;
    var nRound = roundOff(nTotalMB,2);
    var sRounded;

    // if (nTotal < 1024)
    // {
    //     sRounded = '.'+nRound;
    // }
    // else
    // {
    //     sRounded = nRound;
    // }

    return sRounded + ' MB';
}

function XDDiskFree()
{
    var nUsed = XD_gnQuotaUsed;
    var nTotal = XD_gnQuotaTotal;

    var nFreeMB = (nTotal - nUsed)/1024;
    var nRound = roundOff(nFreeMB,2);
    var sRounded;

    // if (nFreeMB < 1)
    // {
    //     sRounded = '.'+nRound;
    // }
    // else
    // {
    //     sRounded = nRound;
    // }

    return sRounded + ' MB';
}
```

//launch.js

```

/*****
* XDEplorerLaunch: Launch the passed explorer URL in a popup window.
*****/

function XDEplorerLaunch (
    sURL, /*** (I) The URL to open in the popup window
    nHeight, /*** (I) The height of the popup
    nWidth) /*** (I) The width of the popup
    {
        var w =
window.open(sURL,"XDriveExplorer","location=no,toolbar=no,menubar=yes,"+
            "status=no,resizable=no,scrolling=yes,scrollbars=no,"+
            "width="+nWidth+",height="+nHeight);

        /*** make sure the opener knows who the parent is
        if (w.opener == null) w.opener = self;

        /*** focus on the newly created window
        w.focus();
    }

function XDEplorerURL()
{
    return '/cgi-bin/explorer.cgi';
}

function XDDataURL()
{
    return '/cgi-bin/explorer_data.cgi';
}

```

//nav.js

```

// Added by Julie Wang 111999
//
// Function is used with <a href> to pop up another window to show X:drive's
Terms of Service
// page

function toc()
{
var url, window_name;

url="/company/toc.html";
window_name="toc";
window.open(
    url,
    window_name,

'toolbar=no,menubar=no,scrollbars=yes,fullscreen=no,resizable=no,width=650,height=400'
);
return;
}

// Added by Julie Wang 122199
//
// Function is used with <a href> to pop up another window to show a
// sample letter when someone use "Tell A Friend" feature.

function tell_a_friend_sample_email()
{
var url, window_name;

url="/generic_join_sample_email.html";
window_name="toc";
window.open(
    url,
    window_name,

'toolbar=no,menubar=no,scrollbars=yes,fullscreen=no,resizable=no,width=650,height=400'
);
return;
}

// Added by Julie Wang 102699
//
// Function writes the side bar nav. menu/buttons on general HTML pages for
every visitors.

function left_menu()
{
    document.write('<table width=\\"138\\" border=\\"0\\" cellpadding=\\"0\\" cellspacing=\\"0\\">\n');
    document.write('<tr align=\\"left\\" valign=\\"top\\">\n');
    document.write('<td><a href=\\"http://www.xdrive.com\\"><img
src=\\"/graphics/internal/btn_xdrivehome.gif\\" width=\\"138\\" height=\\"19\\"
alt=\\"X:drive Home\\" border=\\"0\\"></a><br><img
src=\\"/graphics/internal/divider.gif\\" width=\\"138\\" height=\\"5\\"
alt=\\"Divider\\"></td>\n');
    document.write('</tr>\n');
}

```



```

document.write('<tr align="left" valign="top">\n');
document.write('<td>\n');
document.write('<table width="138" border="0" cellpadding="0" cellspacing="0" background="/graphics/internal/lines.gif">\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font face="Arial, Helvetica, sans-serif" size="2"><a href="/company/new.html">What\'s New</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font face="Arial, Helvetica, sans-serif" size="2"><a href="/company/company.html">About X:drive</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font face="Arial, Helvetica, sans-serif" size="2"><a href="/company/main_download.html">Desktop X:drive</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font face="Arial, Helvetica, sans-serif" size="2"><a href="/affiliates/befree/index.html">X:drive Affiliate</a></font></b></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td width="38"></td>\n');
document.write('<td align="left" valign="middle" width="100"><b><font face="Arial, Helvetica, sans-serif" size="2"><a href="/company/faq.html">FAQ</a></font></b></td>\n');
document.write('</tr>\n');
document.write('</table>\n');

document.write('</td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td><a href="/cgi-bin/signup_form.cgi"></a></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td><br></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td><a href="/freebies/english/freebiesout.html"></a><br></td>\n');
document.write('</tr>\n');

```

```

document.write('<tr align="left" valign="top">\n');
document.write('<td><a href="/company/main_download.html"></a><br></td>\n');
document.write('</tr>\n');
document.write('<tr align="left" valign="top">\n');
document.write('<td><a href="/demo/index.html"></a><br></td>\n');
document.write('</tr>\n');
document.write('</table>\n');

document.close();
return true;
}

```

```

// Added by Martin Hald
function PathRemovePrefix(path)
{
    return path.substring(10,path.length);
}

```

```

// Function that redraws the file explorer

```

```

function show()
{
    var oDocument = FrameObject();

    oDocument.open("text/html");
    oDocument.write("<html>\n");
    oDocument.write("<head>\n");

    oDocument.write("</head>\n");
    oDocument.write('<body BGCOLOR="' + XDBBackgroundColor() + '"
BACKGROUND="' + XDBBackgroundImage() + '">');
    oDocument.write(XD_sNewdoc);

    XDdisplayDiskInfo(oDocument);

    oDocument.write("</body>\n");
    oDocument.write("</html>\n");
    oDocument.close();
    XDrefreshAllButtons();
}

```

```

// parses the XML tree from the top frame and first calls show.

```

```

// This must be called on load of the main page.

```

```

function process(sExtra)

```

```

{
    if (XD_gsAction == '')
    {
        grove = Xparse(XD_gsXML);
    }
}

```

```

//this resets the variables that track how many files and folders
are selected

```

```

//don't reset if we are going into an action
XDresetSelected();

```

```

    }

    // If we have just performed an action that involved a folder then
    // we will open that folder so the user can see the results of the
    // action. To do so we update the old directory listing so that
    // the directory from which the action took place gets opened.
    if (XD_gnSelectedFolderID != '')
    {
        XD_gsPreviousGrove.index[XD_gnSelectedFolderID].attributes.show =
1;
    }

    // Now sync the view of the filesystem between the current and
    // previous views.
    synch(XD_gsPreviousGrove, grove);

    //reset attributes.selected for all items so that blue line does not
get drawn
    XDresetAllSelected();
}

function BuildUpload()
{
    var oDocument = FrameObject();
    XD_gsActionUpload = true;
    HTMLGenericStart(oDocument);

    // var rand_num = parent.createRandomID();
    var rand_num = createRandomID();

    if (XD_gbExtraHelp)
    {
        oDocument.write(XDHelp(XD_gsHelpFileUpload));
    }

    oDocument.write("</TABLE>\n");

    oDocument.write('<p>\n');
    oDocument.write(XDHelp(XD_gsClientAd));
    oDocument.write('</p>\n');

    oDocument.write('<form name="form_upload" method="POST" action="/cgi-
bin/file_save.cgi" onSubmit="return
parent.parent.parent.openUpload(parent.parent.parent.XDCheckFormInput(),\'/cg
i-bin/file_upload_stat.cgi?id=\'+rand_num+\'\',\'window\',(this));"
TARGET="centerview"');

    // oDocument.write('<form name="form_upload" method="POST"
action="/cgi-bin/file_save.cgi" onSubmit="return
(parent.parent.parent.XDCheckFormInput());" TARGET="centerview"');

    oDocument.write(' enctype="multipart/form-data">'+ "\n");
    var results = '';
    results += '<input type="hidden" name="sFolderCurrent" value="'+
XDSelectedFolder() +' ">\n';
    oDocument.write(results);

    oDocument.write('<input type=hidden name=id value=\'+rand_num+\'>');

    oDocument.write('<TABLE cols=2>'+ "\n");

    for (var i=1; i<=5; i++)

```

```

    {
        oDocument.write('<tr><td valign="top" width="30"><FONT face="verdana,
arial, sans" size="-1"><b>' + XD_gsFile + i + ': </b></FONT></td><td><FONT
face="verdana, arial, sans" size="-1"><input type="file"
name="file_to_upload_0' + i + '" size="20"></FONT></td></tr>'+"\\n");
    }
    oDocument.write('</tr>'+"\\n");
    oDocument.write('<tr valign="top"> '+"\\n<td colspan=2>\\n");
    oDocument.write('<center>'+"\\n");
    oDocument.write(XDFormSubmitButtons());
    oDocument.write('</center>'+"\\n");
    oDocument.write('</td>'+"\\n");
    oDocument.write('</tr>'+"\\n");
    oDocument.write('</TABLE>');
    oDocument.write('</body>'+"\\n");
    oDocument.write('</html>'+"\\n");
    oDocument.close();
    XD_gnFrameHeight='85';
    return true;
}

function BuildCreate()
{
    var oDocument = FrameObject();
    HTMLGenericStart(oDocument);

    if (XD_gbExtraHelp)
    {
        oDocument.write(XDHelp(XD_gsHelpCreateFolder));
    }

    oDocument.write('<form name="form_create" action="/cgi-
bin/folder_create.cgi" method="POST" onSubmit="return
parent.parent.parent.XDCheckFormInput();" target="centerview">');
    var results = '';
    results += '<input type="hidden" name="sFolderCurrent" value="' +
XDSelectedFolder() + '">\\n';
    oDocument.write(results);
    oDocument.write('<tr><td valign=center><B>' + XD_gsFolderName + ':</b>');
    oDocument.write('<input type="text" name="sFolderNew" value=""><br>');
    oDocument.write(XDFormSubmitButtons());
    oDocument.write('</td></tr>');

    oDocument.write('</TABLE>');
    oDocument.write('</body>'+"\\n");
    oDocument.write('</html>'+"\\n");
    oDocument.close();
    XD_gnFrameHeight='85';
    return true;
}

function BuildRename()
{
    var oDocument = FrameObject();
    HTMLGenericStart(oDocument);

    if (XD_gbExtraHelp)
    {
        oDocument.write(XDHelp(XD_gsHelpFolderRename));
    }
}

```

```

oDocument.write('<form method="POST" name="form_rename" action="/cgi-
bin/selected_rename.cgi" onSubmit="return
parent.parent.parent.XDCheckFormInput();"');
oDocument.write(' target="centerview" value="'+XDSelected()+'">\n');
var results = '';
results += '<input type="hidden" name="sFolderCurrent"
value="'+XDSelectedFolder()+'">\n';
oDocument.write(results);
oDocument.write('<tr><td valign=center><B>' + XD_gsNewName + ';</b>');
oDocument.write('<input type="hidden" name="sItemCurrent" value="'+
XDSelected() +'">\n');

if (XDProfileEditExtensions)
{
oDocument.write('<input type="text" name="sItemNew" value="" +
XDSelectedThingName()+'">\n');
oDocument.write('<input type="hidden" name="sItemExtension"
value="">\n');
}
else
{
oDocument.write('<input type="text" name="sItemNew"
value="'+XDSelectedThingNameMinusExtension()+'">'+XDSelectedThingNameExtensio
n()+'\n');
oDocument.write('<input type="hidden" name="sItemExtension"
value="'+XDSelectedThingNameExtension()+'">\n');
}
oDocument.write(XDFormSubmitButtons());
oDocument.write('</td></tr>');
oDocument.write('</TABLE>');
oDocument.write('</body>'+""'\n');
oDocument.write('</html>'+""'\n');
oDocument.close();
XD_gnFrameHeight='85';
return true;
}

function BuildDelete()
{
var oDocument = FrameObject();
HTMLGenericStart(oDocument);

var pathToFile = XDSelected();
var lastSlash = pathToFile.lastIndexOf('/');
var file = pathToFile.substring(lastSlash+1,pathToFile.length);

if (XD_gbExtraHelp)
{
oDocument.write(XDHelp(XD_gsHelpDelete));
}
oDocument.write('<form name="form_delete" action="/cgi-
bin/selected_delete.cgi" method="POST" onSubmit="return
parent.parent.parent.XDCheckFormInput();" target="centerview">');
var results = '';
results += '<input type="hidden" name="sFolderCurrent" value="'+
XDSelectedFolder() +'">\n';
oDocument.write(results);
oDocument.write('<tr><td valign=center><B>' + XD_gsSureDelete + ' ' +
file + ';</b>');
oDocument.write('<input type="hidden" name="sItemCurrent" value="" +
XDSelected() + '"><br>');
oDocument.write('<input type="hidden" name="sFolderCurrent" value="" +
XDSelectedFolder() + '"><br>');

```

```

oDocument.write(XDFormSubmitButtons());

oDocument.write('</td></tr>');
HTMLGenericEnd(oDocument);
XD_gnFrameHeight='85';
return true;
}

```

```

function BuildExplorer (grove, sStartDirectory)
{
    var returnValue = true;

    if (XD_gsAction == 'Upload')
    {
        returnValue = BuildUpload();
    }
    else if (XD_gsAction == 'Create')
    {
        returnValue = BuildCreate();
    }
    else if (XD_gsAction == 'Rename')
    {
        returnValue = BuildRename();
    }
    else if (XD_gsAction == 'Delete')
    {
        returnValue = BuildDelete();
    }
    else
    {
        var result = '';
        var nDepth = -2;

        result += '<TABLE compact border=0 cellpadding=4
width="" + XD_gnExplorerTableWidth + "">\n';
        result += XDFormSubmitButtons(1);

        result += "<tr><th align=\"left\">" + XDEplorerFont() + '<font
size="2">' + XDPossessive(XD_gsFirstName + ' ' + XD_gsLastName) + " X:drive
<BR>"
        +XDDiskTotal()+" "+XD_gsCapacity+", "
        +XDDiskFree()+" "+XD_gsRemaining
        +"</th><th align=\"left\">" +
        XDEplorerFont()+'<font size="2">' + XD_gsSize + "</th><th
align=\"left\">" +
        XDEplorerFont()+'<font size="2">' + XD_gsLastModified +
"</th></tr>\n";
        result += dotag(grove, sStartDirectory, nDepth);
        result += "</TABLE>\n";
        XD_sNewdoc = result;

        show();

        //johngaa 11/22/99
        //Highlight bug fix
        if (XD_gsXOffset || XD_gsYOffset)
        {
            XD_goFrameFileExplorer.scrollTo(XD_gsXOffset,XD_gsYOffset);
        }
        //end of johngaa bug fix
    }
}

```

```

    }
    return returnValue;
}

function XDPossessive(name)
{
    var length = name.length;
    var lastChar = name.charAt(length-1);
    var possessive=name + "'s";
    if (lastChar == 's')
    {
        possessive = name + "'";
    }
    return possessive;
}

function XDExplorerFont()
{
    return '<font face="verdana, arial, sans">';
}

// constructs the HTML from the file explorer from the parsed XML
function dotag(tag, path, nDepth)
{
    path += '/' + tag.name;

    var result = '';
    var sCellColor = new String();
    var sIconImage = new String();
    var sFolderPointer = new String();
    var fileSize = new String();
    var fileString = new String();

    var sDate;    // The last modified date and time stamp

    //johngaa 11/23/99
    //highlight netscape bug fix
    // var sFlipFunction = new String('parent.parent.parent.flip(' +
tag.uid + ')');
    if (navigator.appName == "Netscape")
    {
        var sFlipFunction = new String('parent.parent.parent.flip(' +
tag.uid + ',window.pageXOffset,window.pageYOffset)');
    }
    else
    {
        var sFlipFunction = new String('parent.parent.parent.flip(' +
tag.uid + ',document.body.scrollLeft,document.body.scrollTop)');
    }

    //johngaa original 11/22/99
    //highlight netscape bug fix
    //var sSelectToggleFunction = new
String('parent.parent.parent.XDselectToggle(' + tag.uid + ')');
    if (navigator.appName == "Netscape")
    {
        var sSelectToggleFunction = new
String('parent.parent.parent.XDselectToggle(' + tag.uid +
',window.pageXOffset,window.pageYOffset)');
    }
}

```

```

else
{
    var sSelectToggleFunction = new
String('parent.parent.parent.XDselectToggle(' + tag.uid +
',document.body.scrollLeft,document.body.scrollTop)');
}
//end of johngaa bug fix

// If the object is selected,
// then add it to the selected arrays
// and up the selected counts
// and set the cell color to selected

//set background color of the cells depending on status:  selected,
move or at rest
if (tag.attributes.selected)
{
    XD_gnSelectedCount=1;
    sCellColor =XD_gsSelectedColor;
    XD_gsSelectedList += PathRemovePrefix(path) + '+';

    if (tag.attributes.folder)
    {
        XD_gnSelectedFolderCount=1;
        XD_gnSelectedFolderID = tag.uid;
        XD_gsSelectedFolderList += PathRemovePrefix(path) + '+';
    }
    else
    {
        XD_gnSelectedFileCount=1;
    }
}
else if (tag.attributes.move)
{
    // ELSE IF, it is set to move,
    // Then change the colors and
    sCellColor = XD_gsMoveSelectedColor;
}
else
{
    // ELSE, set the cell color to not selected
    sCellColor = XD_gsNotSelectedColor;
}

if (tag.attributes.folder)
{
    // SET special graphics and links for folder.
    nDepth++;

    if (tag.attributes.show)
    {
        if (tag.attributes.move)
        {
            // The folder is open
            sFolderPointer = '<IMG SRC="' +
XD_gimgOpenFolderPointer + '" BORDER="0">\n';
            sIconImage = '<IMG SRC="' + XD_gimgOpenFolder + '"
BORDER="0" ALIGN="absmiddle" '+'\n\t'+ 'HSPACE="2" VSPACE="0" HEIGHT="16"
WIDTH="16">';
        }
        else
        {

```



```

        sFolderPointer = '<A HREF="javascript:' +
sFlipFunction + ';"><IMG SRC="' + XD_gimgOpenFolderPointer + '"
BORDER="0"></A>\n';
        sIconImage = '<IMG SRC="' + XD_gimgOpenFolder + '"
BORDER="0" ALIGN="absmiddle" '+'\n\t" + 'HSPACE="2" VSPACE="0" HEIGHT="16"
WIDTH="16">';
    }
    else
    {
        sFolderPointer = '<A HREF="javascript:' + sFlipFunction +
';"><IMG SRC="' + XD_gimgClosedFolderPointer + '" '+'\n\t" + '
BORDER="0"></A>\n';
        sIconImage = '<IMG SRC="' + XD_gimgFolder + '" BORDER="0"
ALIGN="absmiddle" '+'\n\t" + 'HSPACE="2" VSPACE="0" HEIGHT="16" WIDTH="16">';
    }
    else
    {
        // This is a file and not a folder so show a FILE icon and do not
show any + or -
        sFolderPointer = ExplorerBlankFolderPointer();
        sIconImage = '<IMG SRC="' + XD_gimgFile + '" BORDER="0"
ALIGN="absmiddle" '+'\n\t" + 'HSPACE="4" VSPACE="0">';
    }

    if (tag.attributes.size)
    {
        // SET file size indicator is attribute is present
        fileSize = XDExplorerFont()+tag.attributes.size+'k';
    }
    else
    {
        fileSize = '&nbsp;';
    }

    if (tag.attributes.lastModified)
    {
        sDate = tag.attributes.lastModified;
    }
    else
    {
        sDate = '&nbsp;';
    }

    if (tag.attributes.move)
    {
        fileString= sIconImage;
    }
    else
    {
        fileString = '<A HREF="javascript:' + sSelectToggleFunction +
';">' + '\n' + sIconImage;
    }

    if ((tag.attributes.folder) || (!XDAction('Move')) ||
(tag.attributes.move))
    {
        // ONLY show IF it is (a folder or not in moving)
        // OR the object is question is being moved.
        result += '<A NAME="' + tag.name + '"></A><TR>';
    }

```

```

        result += '<TD BGCOLOR="' + sCellColor + '"
valign="absmiddle"><p>';
        result += "\n";
        result += "\n";
        result += _indent(nDepth);
        result += sFolderPointer;
        result += fileString;
        result += XDEplorerFont();
        result += '<FONT SIZE="2">';
        result += tag.name;
        result += '</A></TD>';
        result += "\n";
        result += "\n";
        result += '<TD BGCOLOR="' + sCellColor + '"
valign="absmiddle"><p><FONT SIZE="2">' + fileSize + '</FONT></TD>';
        result += '<TD BGCOLOR="' + sCellColor + '"
valign="absmiddle"><p><FONT SIZE="2">';
        result += XDEplorerFont();
        result += sDate;
        result += "</FONT></td>\n";

        result += '</TR>';
        result += "\n";
    }

    if (tag.attributes.show)
    {
        for (var i = 0; i < tag.contents.length; i++)
        {
            if (tag.contents[i].type == "element")
            {
                // To sort we simply recursively call ourselves with
                // in the sort order
                result += dotag(tag.contents[i], path, nDepth);
                result += "\n";
            }
        }
    }

    return result;
}

```

```

function ExplorerBlankFolderPointer ()
{
    return '<IMG SRC="/images/explorer/fnot.gif" WIDTH=15 HEIGHT=15
BORDER=0>\n';
}

```

// returns a true if the tag has any children that are selected

```

function XDopenChild(tag, children)
{
    var result = false;

    if (children)
    {
        if ((tag.attributes.selected) || (tag.attributes.move))
        {
            //added so user can close folder if items are selected
            //deselects item in folder if folder is closed
            tag.attributes.selected=false;
        }
    }
}

```

```

        //original
        return true;
    }
}

for (var i = 0; i < tag.contents.length; i++)
{
    if (tag.contents[i].type == "element")
    {
        if (XDopenChild(tag.contents[i], 1))
        {
            //added so user can close folder if items are selected
            //deselects item in folder if folder is closed
            grove.index[i].attributes.selected = false;
            return false;

            //original
            //return true;
        }
    }
}

return result;
}

function _indent (count)
{
    var spaces = '';
    for (i=0; i<=count; i++)
    {
        spaces += '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&';
    }
    return spaces;
}

// This is called when a item name is clicked,
// either flipping it open or closed

//original johngaa 11/23/99
//highlight netscape bug fix
//original function flip(id)
function flip(id,xoffset,yoffset)
{
    //johngaa 11/23/99
    //highlight netscape bug fix
    XD_gsYOffset = yoffset;
    XD_gsXOffset = xoffset;

    //end of johngaa add
    XDresetSelected();

    // before closing, check to see if it has selected children.
    // If child is selected, then do not allow to close dir.
    if (!XDopenChild(grove.index[id], 0))
    {
        if (grove.index[id].attributes.show == 1)
        {
            grove.index[id].attributes.show = 0;
        }
        else

```

```

        }
        grove.index[id].attributes.show = 1;
    }

    BuildExplorer(grove,XD_gsRootDirectory);
}

// This is called when an item icon is clicked, causing
// it to toggle between selected and not selected

//original johngaa 11/22/99
//highlight netscape bug fix
//function XDselectToggle(id)
function XDselectToggle(id,xoffset,yoffset)
{

    //johngaa 11/22/99
    //highlight bug fix
    XD_gsYOffset = yoffset;
    XD_gsXOffset = xoffset;
    //end of johngaa bug fix

    // Martin to solve bug where we log in and we get the error grove.index
    // is not an object
    if (! grove.index)
    {
        return;
    }

    if (id>=0)
    {
        XDresetSelected();

        if (grove.index[id].attributes.selected)
        {
            grove.index[id].attributes.selected = false;
        }
        else
        {
            XDresetAllSelected();
            XD_gnSelectedCount++;
            grove.index[id].attributes.selected = true;
            if (grove.index[id].attributes.folder)
            {
                XD_gnSelectedFolderCount++;
                grove.index[id].attributes.show = 1;
            }
            else
            {
                XD_gnSelectedFileCount++;
            }
        }
    }
    else
    {
        XDresetAllSelected();
    }

    //if this is the page generated directly after a login
    //make XDrive the default and select it
    //then reset variable so we no longer select Xdrive as the default

```

```

    if (XD_gnLogin==1)
    {
        grove.index[0].attributes.selected=true;
        XD_gnLogin=0;
    }

    //this is called every time the file explorer changes
    //including creates, moves, deletes and renames
    //use a setTimeout for NS on NT because otherwise the
    //browser crashes if there is no wait period
    setTimeout("BuildExplorer(grove,XD_gsRootDirectory)",50);
    //BuildExplorer(grove,XD_gsRootDirectory);

}

// function to check to see if the root is selected
function XDRootSelected()
{
    if (grove.index[0].attributes.selected)
    {
        return true;
    }
    return false;
}

// This sets a selection to a value
function XDselect (id,value)
{
    // Martin to solve bug where we log in and we get the error grove.index
    // is not an object
    if (!grove.index)
    {
        return;
    }

    if (grove.index[id].attributes.folder)
    {
        grove.index[id].attributes.selected = true;
        grove.index[id].attributes.show = value;
    }
}

// DeSelects everything if so that only one thing can be selected
// at a time, unless the the multipleSelect checkbox from myFrom
// is selected.
function XDresetAllSelected()
{
    var length = grove.index.length;

    for (var i =0; i < length; i++)
    {
        grove.index[i].attributes.selected = 0;
    }
}

function XDresetAllMovedSelected()
{
    // Martin bug fix -- after the first login could not show X:drive
    if (!grove.index)

```

```

    {
        return;
    }

    var length = grove.index.length;

    var oFrame = XD_goFrameUsageInfo;
    for (var i = 0; i < length; i++)
    {
        grove.index[i].attributes.move = 0;
    }
}

// resets the number of selected, called by both flip and XDselectToggle
function XDresetSelected()
{
    XD_gsSelectedList = '';
    XD_gnSelectedCount = 0;
    XD_gnSelectedFolderCount = 0;
    XD_gnSelectedFileCount = 0;
    XD_gsSelectedFolderList = "";
}

function strip(str)
{
    var A = new Array();

    A = str.split("\n");
    str = A.join("");
    A = str.split(" ");
    str = A.join("");
    A = str.split("\t");
    str = A.join("");

    return str;
}

function entity(str)
{
    var A = new Array();

    A = str.split("&");
    str = A.join("&");
    A = str.split("<");
    str = A.join("<");
    A = str.split(">");
    str = A.join(">");

    return str;
}

function synch (prev_grove, new_grove)
{
    var prev_tag, new_tag, pi, ni;

    if (! prev_grove)
    {
        //set a flag so we know the first time a user logs in
        //there will be no prev_grove in this one case
        //flag is used to show blue bar on XDrive folder only right after
        logging in
        XD_gnLogin=1;
        return;
    }
}

```

```

    }
    //NS4.05 doesn't like this syntax
    //change to new syntax
    //if (! prev_grove.attributes)
        if (prev_grove.attributes!='')
        {
            return;
        }

    if (! new_grove.contents)
    {
        return;
    }

    if (prev_grove.attributes.show)
    {
        pi = 0;

        for (var ni = 0; ni < new_grove.contents.length; ni++)
        {
            if (new_grove.contents[ni].type == "element")
            {
                if (prev_grove.contents[pi])
                {
                    prev_tag = prev_grove.contents[pi];
                }

                if (new_grove.contents[ni])
                {
                    new_tag = new_grove.contents[ni];
                }

                if ((prev_tag) && (new_tag))
                {
                    if (prev_tag.name == new_tag.name)
                    {
                        // Make sure the contents for this object
                        // to avoid javascript "has no
                        // properties" errors.
                        if (prev_grove.contents[pi])
                        {
                            new_grove.contents[ni].attributes =
                                prev_grove.contents[pi].attributes;
                        }
                    }
                    else if (prev_tag.name > new_tag.name)
                    {
                        pi++;
                    }
                    else
                    {
                        ni++;
                    }
                }
            }
            synch(prev_grove.contents[pi],
                new_grove.contents[ni]);
            pi++;
        }
    }
}

```


//saveToXdrive.js

```

var win = external.menuArguments;

ExtMen = external.menuArguments;
ExtMenTag = ExtMen.event.srcElement;
ExtMenDoc = ExtMen.document;

var url;

function findAnchor(el) {
    while ((el!=null) && ((el.tagName!="A") && (el.href!="")))
        el = el.parentElement;
    return el;
}

function findUrl() {
    var re;
    var IMGinsideLink = false;

    //alert("Tag name is " + ExtMenTag.tagName);

    switch ( ExtMenTag.tagName ) {
        // if a "LINK", return the link's URL
        case "A" :
            url = ExtMenTag.href;
            break;

        case "TD":
            var el = win.document.selection.createRange();
            a = findAnchor(el.parentElement(0));
            if (a != null)
            {
                url = a.href;
            }
            break;

        // if it was an image, then this gets complicated:
        case "IMG" :

            // check all links to make sure we aren't in one:
            for ( count = 0; count < ExtMenDoc.links.length; count++ )
                if ( ExtMenDoc.links( count ).contains( ExtMenTag ) ) {
                    IMGinsideLink = true;
                    break;
                }

            // if none was found, return the image URL:
            if ( !IMGinsideLink )
                url = ExtMenTag.src;
            else {

                url = ExtMenDoc.links( count ).href;
            }
            break;
    }
}

```

WO 01/33381

```

default:
    url = ExtMenDoc.href;
    break;
}

// Replace "."
re = /%2e/g;
url = url.replace(re, ".");

// Replace ":"
re = /%3A/g;
url = url.replace(re, ".");

// See if from hotfiles ZD-Net
if (url.indexOf("hotfiles.zdnet") != -1)
{
    var startIndex;
    var endIndex;

    startIndex = url.indexOf("refresh_url=");
    if (startIndex != -1)
    {
        startIndex += 12;
        endIndex = url.indexOf("&", startIndex);

        if (endIndex != -1)
        {
            url = url.substring(startIndex, endIndex);
        }
    }
}

// see if from "download.com" C-Net
else if (url.indexOf("download.com") != -1)
{
    var indexHttp;
    var indexFtp;

    indexHttp = url.lastIndexOf("http://");
    indexFtp = url.lastIndexOf("ftp://");
    index = indexHttp;
    if (indexFtp > indexHttp)
        index = indexFtp;

    //alert( "index is " + index );

    if (index > 0)
    {
        var tempUrl;

        tempUrl = url.substr(index);
        url = tempUrl;
    }
}

}

findUrl();

//alert("begin");
//alert(url);

```

WO 01/33381

// Call X:Drive to perform actual copy

PCT/US00/30536

xd_skip(url);

//secure_login.js

```
//
//  Written 12/1/99
//  Description:
//    Allow users to login securely from the start
//
//

function getState()
{
    //
    //return the value of the checked item
    //called by checkSubmit
    //
    var state;
    if (document.Login.bSecurity[0].checked)
    {
        state = document.Login.bSecurity[0].value;
    }
    else
    {
        state = document.Login.bSecurity[1].value;
    }
    return state;
}

function checkSubmit()
{
    //
    //  checks if secure toggle button is pressed or not
    //  if it is don't allow the submission of the current
    //  form but submit the secureLogin form
    //
    if (getState() == "on")
    {
        document.secureLogin.user.value = document.Login.user.value;
        document.secureLogin.pass.value = document.Login.pass.value;
        document.secureLogin.submit();
        return false;
    }
    else
    {
        return true;
    }
    return false;
}

function writeForm()
{
    //
    //  creates a the secure form
    //
    var fullHostName = XDGetFullHostName();
    var cgiAction = "https://" + fullHostName + "/cgi-bin/login.cgi";
    var formStr;

    formStr = "<form name=\"secureLogin\" method=\"post\" action=\"" +
    formStr += cgiAction;
    formStr += ">";
}
```

```
formStr += "<input type=\"hidden\" name=\"user\" value=\"\">";  
  
formStr += "<input type=\"hidden\" name=\"pass\" value=\"\">";  
formStr += "<input type=\"hidden\" name=\"bSecurity\"  
value=\"on\">\n</form>";  
document.writeln(formStr);
```

```
}
```

```
function clickSecureState()  
{  
    var templ = new String(document.location);  
    var start = -1;  
    start = templ.indexOf("https");  
    if (start != -1)  
    {  
        if (document.Login.bSecurity[0].value == "on")  
        {  
            document.Login.bSecurity[0].click();  
        }  
        else  
        {  
            document.Login.bSecurity[1].click();  
        }  
    }  
}
```

```
}
```

//skip.js

```

//*****
// xd_skip: Popup a skip the download window for the X:Drive skip
// the download service.
//
// Inputs:
//   file_url   : the absolute URL of the file to fetch
//   file_name  : the name to call the stored file
//   file_size  : the file size in KB
//
// Outputs:
//   none
//*****

var skipPartner;
var skipLanguage;
var height = 200;
var width = 575;

function xd_change_location (url)
{
    document.location=url;
}

function xd_skip(file_url,file_name,alt_url,catid,gid,sid,langauge,partner)
{
    var base_url = "http://www.xdrive.com/cgi-bin/skip_the_download.cgi";

    if (! file_name || file_name.length == 0)
    {
        var ii;
        for (ii=0; ii<= file_url.length; ii++)
        {
            if (file_url.charAt(ii) == '/')
            {
                file_name = '';
            }
            else
            {
                file_name = file_name + file_url.charAt(ii);
            }
        }
    }

    var params = "FILEURL=" + escape(file_url) +
        "&FILENAME=" + escape(file_name) +
        "&ALTURL=" + escape(alt_url);

    if (langauge) {
        skipLanguage = langauge;
    }

    if (partner) {
        skipPartner = partner;
    }

    if (skipPartner)

```

```
{
    params = params + "&STDPARTNER=" + escape(skipPartner);
}

if (skipLanguage)
{
    params = params + "&LANG=" + escape(skipLanguage);
}

if (catid)
{
    params = params + "&CATID=" + escape(catid);
}

if (gid)
{
    params = params + "&GID=" + escape(gid);
}

if (sid)
{
    params = params + "&SID=" + escape(sid);
}

if(skipPartner == 'cnet')
{
    height = 235;
    width = 600;
}

url = base_url + "?" + params;
var d = new Date();
var name = d.getTime();

window.open
(
    url,
    name,
    'toolbar=no,menubar=no,scrollbars=no,fullscreen=no,resizable=no,width=' +
width + ',height=' + height
);

return;
}
```

//skipthedownload.js

```

<SCRIPT LANGUAGE="JavaScript"
SRC="http://www.xdrive.com/js/skip.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" defer>

var win = external.menuArguments;

ExtMen = external.menuArguments;
ExtMenTag = ExtMen.event.srcElement;
ExtMenDoc = ExtMen.document;

var url;

function findAnchor(el) {

    while ((el!=null) && ((el.tagName!="A") && (el.href!="")))
        el = el.parentElement;
    return el;

}

function findUrl() {
    var re;
    var IMGinsideLink = false;

    //alert("Tag name is " + ExtMenTag.tagName);

    switch ( ExtMenTag.tagName ) {
        // if a "LINK", return the link's URL
        case "A" :
            url = ExtMenTag.href;
            break;

        case "TD":
            var el = win.document.selection.createRange();
            a = findAnchor(el.parentElement(0));
            if (a != null)
            {
                url = a.href;
            }
            break;

        // if it was an image, then this gets complicated:
        case "IMG" :

            // check all links to make sure we aren't in one:
            for ( count = 0; count < ExtMenDoc.links.length; count++ )
                if ( ExtMenDoc.links( count ).contains( ExtMenTag ) ) {
                    IMGinsideLink = true;
                    break;
                }

            // if none was found, return the image URL:
            if ( !IMGinsideLink )
                url = ExtMenTag.src;
            else {

```



```

        url = ExtMenDoc.links( count ).href;
    }
    break;

default:
    url = ExtMenDoc.href;
    break;
}

// Replace "."
re = /%2e/g;
url = url.replace(re, ".");

// Replace ":"
re = /%3A/g;
url = url.replace(re, ".");

// See if from hotfiles ZD-Net
if (url.indexOf("hotfiles.zdnet") != -1)
{
    var startIndex;
    var endIndex;

    startIndex = url.indexOf("refresh_url=");
    if (startIndex != -1)
    {
        startIndex += 12;
        endIndex = url.indexOf("&", startIndex);

        if (endIndex != -1)
        {
            url = url.substring(startIndex, endIndex);
        }
    }
}

// see if from "download.com" C-Net
else if (url.indexOf("download.com") != -1)
{
    var indexHttp;
    var indexFtp;

    indexHttp = url.lastIndexOf("http://");
    indexFtp = url.lastIndexOf("ftp://");
    index = indexHttp;
    if (indexFtp > indexHttp)
        index = indexFtp;

    //alert( "index is " + index );

    if (index > 0)
    {
        var tempUrl;

        tempUrl = url.substr(index);
        url = tempUrl;
    }
}

}

findUrl();

```

```
//alert("begin");  
//alert(url);  
  
// Call X:Drive to perform actual copy  
  
xd_skip(url);  
  
</script>
```

//submit.js

```

/*****
*
* Submit.JS: This javascript class is for all the actions associated with
* buttons. This class may either open a new window or submit an existing
* form for server parsing.
*****/

/*****
* XDCheckFormInput() - check upload/rename/create input.
* if there are errors, give then alert. if not, submit
*****/

function XDCheckFormInput()
{
    //make sure user is not allowed to upload a blank file
    if (XD_gsAction == 'Upload')
    {
        sFormName = XD_goFrameFileExplorer.document.form_upload;
        if (sFormName.file_to_upload_01.value == '')
        {
            alert(XD_gsAlertUploadEmptyFile);
            return false;
        }
    }
    //make sure user cannot create a blank file
    else if (XD_gsAction == 'Create')
    {
        sFormName = XD_goFrameFileExplorer.document.form_create;
        if (sFormName.sFolderNew.value == '')
        {
            alert(XD_gsAlertCreateEmptyFile);
            return false;
        }
    }
    else if (XD_gsAction == 'Rename')
    {
        sFormName = XD_goFrameFileExplorer.document.form_rename;
        //do not allow user to rename file the same name it already has
        //find just the file name to compare to what was input
        var lastSlash=sFormName.sItemCurrent.value.lastIndexOf('/');

        //if this is a folder of user may edit file extensions, use this
        code
        if ((parent.parent.XDProfileEditExtensions) ||
        (XD_gnSelectedFileCount==0))
        {
            the
            //allow user to edit extensions so check everything after
            //last slash
            var
            fileName=sFormName.sItemCurrent.value.substring(lastSlash+1,sFormName.sItemCu
            rrent.value.length);

            if (fileName == sFormName.sItemNew.value)

```

```

        {
            alert(XD_gsAlertRenameSameName);
            return false;
        }
    }
    else
    {
        //do not allow user to edit extensions so need to find last
        '..' as well
        var lastDot=sFormName.sItemCurrent.value.lastIndexOf('.');
        var
        fileName=sFormName.sItemCurrent.value.substring(lastSlash+1,lastDot);

        if (fileName == sFormName.sItemNew.value)
        {
            alert(XD_gsAlertRenameSameName);
            return false;
        }

        //check to see if user is trying to name the file/folder nothing
        //give em an error message if so
        if (sFormName.sItemNew.value == '')
        {
            alert(XD_gsAlertRenameNothing);
            return false;
        }
    }
    else { }

    XD_gsAction = '';
    //sFormName.submit();
    return true;
}

function XDSubmitView (sFormName) {
    // Always start by checking the status and if the status is not active
    then
    // return and do not perform any actions.
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_VIEW))

    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_VIEW)) {
        return false;
    }

    var sFixed = '';
    var sFileName = XDSelected();

    for (i = 0;i <= sFileName.length;i++) {
        if (sFileName.charAt(i) == ' ') {
            sFixed += '+';
        }
        else {
            sFixed += sFileName.charAt(i);
        }
    }

    // URL encode/escape string
    sFixed = escape(sFixed);

    var sURL = '/cgi-bin/file_load.cgi/'+sFixed+'?sFileCurrent=' + sFixed +
    "&source=www.fileExplorer.view";
}

```

```

    XDReaderShow(sURL, 400, 400);
    return true;
}

// Justin's upload status stuff.
function openUpload(form_check, url, name, f) {
    if (! form_check) {
        return false;
    }

    var form_length = f.length;
    var cnt = 0;

    for(var i = 0; i < f.length; i++) {
        var e = f.elements[i];

        if ( (e.type == "file") && (e.value.length > 0) ) {
            cnt++;
        }
    }

    var amp_nof = "&nof=";
    url += amp_nof + cnt;

    msgWindow =
window.open(url, name, 'width=350,height=190,toolbar=no,resize=no,scrollbars=no
');

    return true;
}

function createRandomID () {
    substr_rand_num = new String(Math.random());

    return substr_rand_num.substring(2,14);
}

function XDSubmitDownload ()
{
    // Always start by checking the status and if the status is not
    // active then return and do not perform any actions.
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_DOWNLOAD))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_DOWNLOAD))
    {
        return false;
    }

    var sFileName = XDSelected();
    var oDocument = XD_goFrameData.document;
    var sExtraPath;

    sExtraPath = '/' + sFileName;

    HTMLGenericStart(oDocument);
    oDocument.write('<form name="form_download" target="userData"
method="POST" action="/cgi-bin/file_load.cgi' + sExtraPath + '"
enctype="multipart/form-data">' + "\n");
    oDocument.write('<input type="hidden" name="sFileCurrent"
value="' + sFileName + '">');
    oDocument.write('<input type="hidden" name="mime" value="download">');
    oDocument.write('<input type="hidden" name="source"
value="www.fileExplorer.download">');
}

```

```

        oDocument.write('<input type="hidden" name="sFolderCurrent" value="' +
XDSelectedFolder() + '>');
        //johngaa test add 12/2/99
        oDocument.write('</form>');
        //end of johngaa add
        HTMLGenericEnd(oDocument);
        oDocument.forms[0].submit();
        return true;
    }

function XDSubmitNewFolder(sFormName)
{
    var sNewFolderName = prompt(XD_gsRenamePrompt);
    XDFormSetGeneric(sFormName);
    XDFormSetFolderNew(sFormName, sNewFolderName);
    sFormName.submit();
}

/*****
 * XDItemDelete: Delete an item (no prompting here)
*****/

function XDItemDelete()
{
    var sFileName = XDSelectedList();
    XDFormSetGeneric(sFormName);
    XDFormSetThingName(sFormName);
    sFormName.submit();
}

/*****
 * XDSubmitDelete: Verify they can delete the selected item and then
 * redirect to a web page that will prompt them to delete.
*****/

function XDSubmitDelete(sFormName)
{
    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertDeleteFolder);
        return false;
    }
    location = "delete_prompt.html";
    return true;
}

function XDBufferChange(sFormName, sType)
{
    // We popup a new window for them to select a folder from
    XDFormSetBufferAction(sFormName, sType);
    parent.parent.XDopenFolderSelectWindow();
    XDFormSetSelectedFiles(sFormName);
}

function XDSubmitBufferChange (sFolderTo)
{
    // This method is being access across frames so we cannot easily pass
    the form name
    // so instead we set a variable equal to what the object would have
    been.
    sFormName = window.frames[XD_gsControlFrame].document.form_buffer;

```

```

        XDFormSetGeneric(sFormName);
        XDFormSetFolderNew(sFormName,sFolderTo);
        sFormName.submit();
    }

function XDSubmitMove(sFormName)
{
    XDFrameMove();
    BuildExplorer(grove,XD_gsRootDirectory);
}

/*****
 * XDPopupShow: Show a popup browser
 *****/

function XDReaderShow(sURL, nHeight, nWidth) {
    nWidth = 500;
    nHeight = 600;

    var r = window.open(sURL,"reader","location=no,toolbar=no,menubar=no,"+
        "status=no,resizable=yes,scrolling=yes,scrollbars=yes,"+
        "width="+nWidth+",height="+nHeight);

    /*** make sure the opener knows who the parent is
    if (r.opener == null) r.opener = self;

    /*** focus on the newly created window
    //r.focus();
}

function FrameObject()
{
    return XD_goFrameFileExplorer.document;
}

function HTMLGenericStart (oDocument)
{
    oDocument.open("text/html");
    oDocument.write('<html>');
    oDocument.write("<head>\n");
    oDocument.write("<link rel=stylesheet href='/css/style_back.css'
type='text/css'>\n");
    oDocument.write("</head>\n");
    oDocument.write('<body background="' + XDBackgroundImage() + '"
bgcolor="' + XDBackgroundColor() + '">'+"\n");
    oDocument.write('<table><tr>');

/*****
 * HTMLGenericEnd:
 *****/

function HTMLGenericEnd (oDocument)
{
    oDocument.write('</table>');
    oDocument.write('</body>'+"\n");
    oDocument.write('</html>'+"\n");
    oDocument.close();
}

```

```

function XDBuildForm()
{
    var form = '';
    var sSubmitButton = '/images/submit.gif';

    if (XDAction('Move'))
    {
        form += '<form name="form_buffer" action="/cgi-bin/buffer_paste.cgi" +
            ' method="POST" target="centerview"' +
            ' parent.parent.parent.XRReset();">' + "\n";
        form += '<input type="hidden" name="sFile" value="' + XD_gsMoveSelectedList + '">';
        sSubmitButton = '/images/move.gif';
        XD_gnFrameHeight = '40';
    }

    form += '<input type="hidden" name="sFolderCurrent" value="' + XDSelectedFolder() + '">';
    form += '<input type="hidden" name="type" value="move">';
    form += '<input type="hidden" name="sItemCurrent" value="">';
    form += '<input type="hidden" name="sFolderNew" value="">';

    form += '<p><INPUT TYPE="button" VALUE="" + XD_gsButtonSubmit + '
onClick="parent.parent.parent.XDSetMoveForm(document.forms[0]);">'+
        '<INPUT TYPE="button" VALUE="" + XD_gsButtonCancel + '
onClick="parent.parent.parent.XRReset();
parent.parent.parent.XDRefreshExplorer();">' +
        '</td>';
    form += '</form>';

    return form;
}

function XDSetMoveForm (oForm)
{
    oForm.sItemCurrent.value = XDSelectedToMove();
    oForm.sFolderNew.value = XDSelectedFolder();

    // adding check for target folder
    if (XD_gsSelectedFolderList.length > 0)
    {
        //check to see if the user is attempting to move the file into
        //the folder it is already in - can't do that
        var slash=oForm.sItemCurrent.value.lastIndexOf("/");
        var fileDirectory=oForm.sItemCurrent.value.substring(0,slash);

        if (oForm.sFolderNew.value == fileDirectory)
        {
            alert(XD_gsAlertMoveSameFolder);
        }
        else
        {
            // makes sure that the target is not the same as
            // source
            if (oForm.sFolderNew.value == oForm.sItemCurrent.value)
            {
                alert(XD_gsAlertNoTargetFolder);
            }
            else
            {

```



```

the file      //call reset and submit form only if they can actually move
              //else they only get the dialog warning box
              XDReset();
              oForm.submit();
            }
          }
        else
        {
            alert(XD_gsAlertNoTargetFolder);
        }
    }

function XDFormSubmitButtons (generic)
{
    var HTMLString = '';
    var FormString = '';
    var TotalString = '';

    // Grab the appropriate HTML
    if (XDAction('Move'))
    {
        if (XD_gbExtraHelp)
        {
            HTMLString = XDHelp(XD_gsHelpMoveHTML);
        }

        FormString = XDBuildForm();
        return HTMLString + "</TD></TR><TR><TD>" + FormString;
    }
    else if (XDAction('Rename'))
    {
        if (XD_gbExtraHelp)
        {
            HTMLString = XDHelp(XD_gsHelpFolderRename);
        }
    }
    else if (XD_gbExtraHelp)
    {
        if (XD_gnSelectedCount > 0 && ! XDRootSelected())
        {
            if (XD_gnSelectedFileCount)
            {
                HTMLString = XD_gsHelpFileSelected;
            }
            else
            {
                HTMLString = XD_gsHelpFolderSelected;
            }
        }
        else
        {
            if (XD_gsFirstTime)
            {
                HTMLString = XD_gsHelpFirstTimeEnter;
            }
            else
            {
                HTMLString = XD_gsHelpEnter;
            }
        }
    }
}

```

```

        // Format the help box
        HTMLString = XDHelp(HTMLString);
    }

    if (! generic)
    {
        var sSubmitButton;
        sSubmitButton = '/images/submit.gif';
        if (XDAction('Rename'))
        {
            sSubmitButton = XD_gsButtonRename;
        }
        else if (XDAction('Upload'))
        {
            sSubmitButton = XD_gsButtonUpload;
        }
        else if (XDAction('Create'))
        {
            sSubmitButton = XD_gsButtonCreate;
        }
        else if (XDAction('Delete'))
        {
            sSubmitButton = XD_gsButtonDelete;
        }

        return '<p><input type="submit" value="'+sSubmitButton+'">\n'+
            '<input type=button value="'+ XD_gsButtonCancel + '"
onclick="'+
            'parent.parent.parent.XRReset(); '+
            'parent.parent.parent.XRRefreshExplorer();">\n</FORM>';
    }

    TotalString = HTMLString + FormString;
    return TotalString;
}

function XDHelp (sHelp)
{
    return '<tr><td height=50 bgcolor="'+ XD_gsExplorerHelpBackgroundColor
+ '" colspan=3 valign=top><FONT FACE="arial, helvetica" size="-1"
color="#666666"><b>' + XD_gsInstructions + '</b>\n' + sHelp + '\n</td></tr>';
}

/*****
* XDFrameShare: Share a file with another user
*****/

function XDFrameShare()
{
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_SHARE))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_SHARE))
    {
        return false;
    }

    var sFile = XDEscapeCharacters(XDSelected());

```

```

frames['centerview'].document.location = '/cgi-
bin/share_a_file.cgi?help=' +
    XD_gbExtraHelp + '&sFileName=' + sFile;
return true;
}

function XDCheck (sName)
{
    return "if (! XDAllowChange("+sName+") {return false;}";
}

function XDSelectedThingName()
{
    var r = '';
    var s = XDSelected();
    for (var i=0; i<s.length;++i)
    {
        var ch=s.charAt(i);
        if (ch == '/')
        {
            r = '';
        }
        else
        {
            r += ch;
        }
    }
    return r;
}

function XDSelectedThingNameMinusExtension()
{
    var r = '';
    var b = false; // found first time
    var s = XDSelectedThingName();
    for (var i=s.length;i>=0;--i)
    {
        var ch=s.charAt(i);
        if (ch == '.' && ! b)
        {
            b = true;
            r = '';
        }
        else
        {
            r = ch + r;
        }
    }
    return r;
}

function XDSelectedThingNameExtension()
{
    var r = '';
    var s = XDSelectedThingName();
    var bFoundDot = false;
    for (var i=0;i<s.length;++i)
    {
        var ch = s.charAt(i);
        if (ch == '.')
        {
            r = '';
            bFoundDot = true;

```

```

    }
    else
    {
        r += ch;
    }
}
if (bFoundDot == true)
{
    return '.'+r;
}
else
{
    return '';
}
}

/*****
 * XDFrameUpload: Refresh the action frame with a form to perform the file
 * upload and set the form values during the HTML creation itself.
*****/

function XDFrameUpload(sCurrentFolder)
{
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_UPLOAD))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_UPLOAD))
    {
        return false;
    }

    XDActionStart('Upload');
    XD_gnFrameHeight = '1';
    frames['centerview'].document.location = XDCenterView();
    return true;
}

function XDFrameFolderNew ()
{
    //if (!XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_NEWFOLDER))
    if (!XDfunctionStatus(XD_TOOLBAR_BUTTON_NEWFOLDER))
    {
        return false;
    }

    XDActionStart('Create');
    XD_gnFrameHeight = '1';
    frames['centerview'].document.location = XDCenterView();
    return true;
}

function XDFrameRename ()
{
    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertRenameFolder);
        return false;
    }
    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_RENAME))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_RENAME))
    {
        return false;
    }
}

```

```

    XDActionStart('Rename');
    XD_gnFrameHeight = '1';
    frames['centerview'].document.location = XDCenterView();
    return true;
}

```

```

function XDFrameDeletePrompt()
{

```

```

    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertDeleteFolder);
        return false;
    }

```

```

    //if (! XDfunctionStatus(parent.parent.XD_TOOLBAR_BUTTON_DELETE))
    if (! XDfunctionStatus(XD_TOOLBAR_BUTTON_DELETE))
    {
        return false;
    }

```

```

    XDActionStart('Delete');
    XD_gnFrameHeight = '1';
    frames['centerview'].document.location = XDCenterView();
    return true;
}

```

```

/*****
* XDsetSelectedToMove: takes all files that are currently selected and sets
* their move attribute
*****/

```

```

function XDsetSelectedToMove(tag)
{

```

```

    if (tag.attributes.selected)
    {
        tag.attributes.selected = 0;
        tag.attributes.move = 1;
    }

```

```

    for (var i = 0; i < tag.contents.length; i++)
    {
        if (tag.contents[i].type == "element")
        {
            XDsetSelectedToMove(tag.contents[i]);
        }
    }
}

```

```

function XDFrameMove()
{

```

```

    if (! XDAllowChange(XDSelected()))
    {
        alert(XD_gsAlertMoveFolder);
        return false;
    }

```

```

    // XXX
    XD_gsMoveSelectedList = XD_gsSelectedList;
    XD_gsSelectedList = "";
    XDsetSelectedToMove(grove);

    XDActionStart('Move');
}

```

```

        XD_gnFrameHeight = '1';
        frames['centerview'].document.location = XDCenterView();

        return true;
    }

/*****
 * XDBrowserDownloadSupported: Returns true if the browser supports the
 *   download button. This includes all Netscape versions and IE 5 or later.
 *****/

function XDBrowserDownloadSupported()
{
    return !((navigator.appName == "Microsoft Internet Explorer") &&
        (parseInt(navigator.appVersion) <= 4 ));
}

function XDProfile(form)
{
    XDProfileEditExtensions = form.elements['bFileExtEdit'].checked;
    XD_gbExtraHelp = form.elements['bExtraHelp'].checked;
    XD_gbMarketing = form.elements['bMarketing'].checked;
    XD_gbNewsletter = form.elements['bNewsletter'].checked;
}

function XDLogout()
{
    var sUrl = '/cgi-bin/logout.cgi';
    parent.parent.location.href = sUrl;
}

/*****
 *
 * XDSelected: Return the currently selected file or folder and remove the
 * plus that appears at then end -- used the separate elements in a multi
 * file/folder list.
 *****/

function XDSelected()
{
    return XD_gsSelectedList.substring(0,XD_gsSelectedList.length-1);
}

function XDSelectedFolder()
{
    alert(XD_gsLengthofFolder + XD_gsSelectedFolderList.length);
    return
XD_gsSelectedFolderList.substring(0,XD_gsSelectedFolderList.length-1);
}

function XDSelectedToMove()
{
    return
XD_gsMoveSelectedList.substring(0,XD_gsMoveSelectedList.length-1);
}

/*****
 * XDCleanupPath: Cleanup the passed path by removing the "/X:drive/" prefix
 * and the + postfix.
 *****/

```

```

function XDPathCleanup(sPath)
{
    var sCopy = sPath;
    sCopy = sCopy.substring(9,sCopy.length)
    //sCopy = sCopy.substring(0,sCopy.length-1);
    return sCopy;
}

function XDDomain ()
{
    baseAddress = java.net.InetAddress.getLocalHost();
    userDomain = baseAddress.getHostName();
    alert(userDomain.toString());
}

function XDXdrive ()
{
    XDDomain();
    return '/cgi-bin/explorer_user_data.cgi';
}

function XDCenterView ()
{
    return '/cgi-
bin/frame_generic.cgi?thtml=centerview.thtml&sFrameHeight=' +
XD_gnFrameHeight;
}

function XDReset ()
{
    XD_gnSelectedCount = 0;
    XD_gnSelectedFileCount = 0;
    XD_gnSelectedFolderCount = 0;
    XD_gsSelectedList = "";
    XD_gnSelectedFolderID = '';
    XD_gsMoveSelectedList = "";
    XD_gsSelectedFolderList = "";
    XD_gsTargetFolder = "";
    ControlsEnabled = true;
    XDresetAllMovedSelected();
    XDActionEnd();
}

function XDAllowChange (sFolder)
{
    if (sFolder == '' || sFolder == ' ' || sFolder == 'public' || sFolder
== 'private')
    {
        return false;
    }
    return true;
}

function XDAction (sAction)
{
    if (XD_gsAction == sAction)
    {
        return true;
    }
    return false;
}

```

```

// Register a new action
function XDActionStart (sAction)
{
    XD_gsAction = sAction;
}

// Clear the current action
function XDActionEnd ()
{
    XD_gsAction = '';
}

function XDRefreshExplorer()
{
    //reset the action before calling this function
    //or the action screen will be drawn
    XDActionEnd();
    XD_gnFrameHeight = '40';
    //also reset if a move has been started but never finished
    XDresetAllMovedSelected();
    frames['centerview'].document.location=XDCenterView();
}

function XDGetButtonFrameHeight(oDocument)
{
    oDocument.open("text/html");
    oDocument.write(XD_goButtonFrameHeight);
    oDocument.close;
}

function XDSetButtonFrameHeight(height)
{
    XD_gnButtonFrameHeight=height;
}

/*****
** XDRefreshBanner: Refresh the banner with a new advertisement.
*****/

function XDRefreshBanner()
{
    if (XDBannerOn())
    {
        frames['banner'].document.location = '/cgi-bin/ads.cgi';
        // WIP: parent 3 twice removed (from the above line)
    }
}

/*****
** XDBannerOn: Return true if we should display the banner.
*****/

function XDBannerOn()
{
    if (XD_gsPartner == 'xdrv')
    {
        return true;
    }
    else
    {
        return false;
    }
}

```



```
function XDTellAFriend()
{
    var sUrl = '/cgi-
bin/tell_a_friend.cgi?numFriends='+XD_gnNumFriendsToTell;
    frames['centerview'].document.location=sUrl;
}

function XDAddSpace()
{
    var sUrl = '/cgi-bin/addspace.cgi?action=intro';
    frames['centerview'].document.location=sUrl;
}

function XDDownloadClient()
{
    var sUrl = '/cgi-bin/download_client.cgi';
    frames['centerview'].document.location=sUrl;
}
```

//uploadStatus.js

<!--

```
function openUpload(form_check, url, name, f) {  
    if (! form_check) {  
        return false;  
    }  
  
    var form_length = f.length;  
    var cnt = 0;  
  
    for(var i = 0; i < f.length; i++) {  
        var e = f.elements[i];  
  
        if ( (e.type == "file") && (e.value.length > 0) ) {  
            cnt++;  
        }  
    }  
  
    var amp_nof = "&nof=";  
    url += amp_nof + cnt;  
  
    msgWindow =  
window.open(url,name,'width=350,height=190,toolbar=no,resize=no,scrollbars=no  
' );  
  
    return true;  
}  
  
function createRandomID () {  
    substr_rand_num = new String(Math.random());  
  
    return substr_rand_num.substring(2,14);  
}  
. //-->
```

//utils.js

```

/*****
* XDFormSetThingName: Set the name for the thing in the passed form.
*****/

function XDFormSetThingName(sFormName)
{
    sFormName.sThingName.value = XDSelectedList();
}

function XDFormSetBufferAction(sFormName,sType)
{
    sFormName.type.value = sType;
}

function XDFormSetFolderCurrent(sFormName)
{
    sFormName.sFolderCurrent.value = XDSelectedFolder();
}

function XDFormSetSelectedFiles (sFormName)
{
    sFormName.sFile.value = XDSelectedList();
}

function XDFormSetFolderNew(sFormName,sFolderNameNew)
{
    sFormName.sFolderNew.value = sFolderNameNew;
}

/*****
* XDFormSetThingOld: Set the old name attribute for the rename CGI.
*****/

function XDFormSetThingOld(sFormName,sThingName)
{
    sFormName.sThingNameOld.value = sThingName;
}

/*****
* XDFormSetThingNew: Set the new name attribute for the rename CGI.
*****/

function XDFormSetThingNew(sFormName,sThingName)
{
    sFormName.sThingNameNew.value = sThingName;
}

function XDFormSetGeneric(sFormName)
{
    XDFormSetFolderCurrent(sFormName);
}

/*****
* XDPopupShow: Show a popup browser
*****/

function XDPopupShow(
    sURL, /*** (I) The URL to open in the popup window
    nHeight, /*** (I) The height of the popup

```

```

nWidth) /*** (I) The width of the popup
{
var w = window.open(sURL, "viewer", "location=no,toolbar=no,menubar=no,"+
    "status=no,resizable=yes,scrolling=yes,scrollbars=no,"+
    "width="+nWidth+",height="+nHeight);

/*** make sure the opener knows who the parent is
if (w.opener == null) w.opener = self;

/*** focus on the newly created window
w.focus();
}

function XDSelectedList()
{
return XD_gsSelectedList;
}

function XDBackgroundColor()
{
return XD_gsExplorerBackgroundColor;
}

function XDBackgroundImage()
{
return XD_gsBackgroundImage;
}

function XDSelectedFolder()
{
return
XD_gsSelectedFolderList.substring(0,XD_gsSelectedFolderList.length-1);
}

/*****
* XDCleanupPath: Cleanup the passed path by removing the "/X:drive/" prefix
* and the + postfix.
*****/
function XDPathCleanup(sPath)
{
var sCopy = sPath;
sCopy = sCopy.substring(9,sCopy.length)
    //sCopy = sCopy.substring(0,sCopy.length-1);
return sCopy;
}

function XDMultiSelect (sValue)
{
if (sValue != 'null' && sValue != "")
{
m_sMultiSelect = sValue;
}
else
{
return m_sMultiSelect;
}
}

function HTMLNavigation ()
{
var sHTML = HTMLStart()
    +'<table width="100%" border="0" cellpadding="0" cellspacing="0">'

```

```

        + '<tr align="left" valign="top" bgcolor="#5EB114">'
        + '<td></td>'
        + '</tr><tr>'
        + '<td></td>'
        + '</tr><tr>'
        + '<td></td>'
        + '</tr></table>'
        + '<a target="toolbar" href="http://www.mit.edu">MIT</a>'
        + '</BODY>\n</HTML>';
    return sHTML;
}

function HTMLStart ()
{
    return "<HTML>\n"
    + '<body bgcolor="#6961AB" topmargin="0" leftmargin="0" marginheight="0"
marginwidth="0" text="#FFFFFF" link="#FFFFFF" {onload}>'
    + "\n";
}

function HTMLEnd ()
{
    return "\n</BODY>\n</HTML>\n";
}

function RedrawToolBar()
{
    var sWindow = 'window.toolbar';
    sWindow.document.write(HTMLStart()+'test'+HTMLEnd());
}

function XDEscapeCharacters (str)
{
    var A = new Array();

    A = str.split("+");
    str = A.join("%2B");

    A = str.split(" ");
    str = A.join("%20");

    A = str.split("%");
    str = A.join("%25");

    A = str.split("&");
    str = A.join("%26");

    return str;
}

```

//verify_lib.js

```

<!-- Begin Hiding from older browsers

/*****
***      Javascript library of functions commonly
***      used in HTML forms.
*****/

validateForm(form)
    attaches to the submit button and takes the
    form as an argument. Validates all the
    fields and will only let the form be submitted
    if all the fields validate.

checkForm()
    attaches to nothing. Is used by the script
    internally to allow compel() to function w/o
    calling alert(), which would cause an infinite
    loop.

requireElements(num)
    attaches to onLoad to initialize the array
    of required fields in the form.

addRequiredElements()
    attaches to nothing. Is used internally to
    construct a array of the names of all the
    required fields in a form. For this to work
    the form needs a "requiredElements" hidden
    input tag. It should be of this format:

    <INPUT TYPE="hidden" NAME="requiredElements" VALUE=" name:email:">

    List the required field names in order that
    they appear in the form. End each name with
    a ':' and lead the whole value with a blank
    space. If this tag is not used, then
    validateRequiredElements will identify a
    missing required field by its number in lieu
    of the name.

compel(textfield)
    attaches to an onBlur event on a textfield.
    This causes focus to be kept on a textfield
    until checkForm() determines that they user
    has filled it out correctly.

required(textfield, num)
    attached to an onBlur of a field is required.
    The number is it's location on the
    required_elements array. i.e.

    <INPUT TYPE="text" NAME="name" onBlur="required(this, 0)">

    This tag declairs "name" as the first
    required field in the form.

validatePhone(textfield)
    attaches to an onChange of a textfield. This
    function validates to true only if the

```

textfield is blank or contains only
0-9, -, (, or)

validateEmail(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains an @

validateDate(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a date in
the format DD-MON-RRRR

validateDate_old(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a date in
the format DD/MM/YY

validateNum(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a number
between -1 and infinity

validateMoney(textfield)
attaches to an onChange of a textfield. This
function validates to true only if the
textfield is blank or contains a number
between with two decimal places (ie 2.56)

confirmDelete(textfield)
attaches to an onClick on a submit button
used as a delete button that you want the
user to confirm before engaging.

reset_b()
attaches to an onClick on a submit button.
If you have a reset button, used confirmDelete
or have a button that needs to override the
validity of the form, this function should
be attached to these buttons to allow them
to function.

*****/

```
emailOK = true;
phoneOK = true;
dateOK = true;
lengthOK = true;
all_numOK = true;
all_moneyOK = true;
deleteOK = true;
yearOK = true;
required_elements = new Array();
required_elements_names = new Array();
blurred = "";
in_required = false;
submitted = false;
var submitCount;
```

```

function validateForm(form) {
  addRequiredElements(form);
  if (!emailOK) {
    alert(XD_gsValidateEmail);
    return false;
  } else if (!phoneOK) {
    alert(XD_gsValidatePhone);
    return false;
  } else if (!dateOK) {
    alert(XD_gsValidateDate);
    return false;
  } else if (!lengthOK) {
    alert(XD_gsValidateLength);
    return false;
  } else if (!all_numOK) {
    alert(XD_gsValidateNumber);
    return false;
  } else if (!all_moneyOK) {
    alert(XD_gsValidateMoney);
    return false;
  } else if (!yearOK) {
    alert(XD_gsValidateYear);
    return false;
  } else if (!deleteOK) {
    return false;
  } else if (!validateRequiredElements()) {
    return false;
  }
}

```

```

function compel (textfield) {
  if (blurred == "") {
    blurred = textfield;
  }
  if (!checkForm()) {
    blurred.focus();
    blurred.select()
  }
  if (checkForm() && !in_required) {
    blurred = "";
  }
}

```

```

function checkForm() {
  if(!emailOK){
    return false;
  } else if(!phoneOK){
    return false;
  } else if(!dateOK){
    return false;
  } else if(!lengthOK){
    return false;
  } else if(!all_numOK){
    return false;
  } else if(!all_moneyOK){
    return false;
  } else if(!yearOK){
    return false;
  } else if(!deleteOK){
    return false;
  } else {
    return true;
  }
}

```



```

    }
}

function requireElements(num) {
    var i;
    for (i=0; i < num; i++) {
        required_elements[i] = false;
    }
}

function addRequiredElements(form) {
    var found = false;
    for (var n=0; n < form.length; n++) {
        if (form.elements[n].name == "requiredElements") {
            found = true;
        }
    }
    if (found) {
        var length = form.requiredElements.value.length;
        var start_index = 0;
        var end_index = 0;
        var num = 0;
        for (var i=0; i < length; i++) {
            var theChar = form.requiredElements.value.charAt(i);
            if (theChar == ":") {
                start_index = end_index + 1;
                end_index = i;
                var string =
form.requiredElements.value.substring(start_index, end_index);
                num = required_elements_names.length;
                required_elements_names[num] = string;
            } // end of if ":"
        } //end for loop
    } // end of found
}

//check to see if the year is a 4-digit value greater than 1900
function validateYear(textfield)
{
    yearOK = true;

    //make sure the file contains only numbers
    for (var n=0; n < textfield.value.length; n++)
    {
        var theChar = textfield.value.charAt(n);
        if ((theChar >= "0") && (theChar <= "9"))
        {
            //do nothing, assume it's still true
        }
        else
        {
            //contains non numeric elements
            yearOK=false;
        }
    }

    if (!yearOK)
    {

```

```

        alert(XD_gsValidateContainNums);
    }

    if (textfield.value < 1900)
    {
        yearOK = false;
        alert(XD_gsValidateGreater1900);
    }

    if (textfield.value.length != 4)
    {
        yearOK = false;
        alert(XD_gsValidateFourDigits);
    }
}

// Checks for a properly formatted email
function validateEmail(textfield)
{
    emailOK = true;

    if ((textfield.value == "") || (textfield.value.indexOf("@") < 0))
    {
        emailOK = false;
        alert(XD_gsValidateEmailFormat);
        return false;
    }
    return true;
}

function required(textfield, num)
{
    var alert_show = false;
    in_required = true;
    if (blurred == "")
    {
        alert_show = true;
        blurred = textfield;
    }

    if(textfield.type == "select-one")
    {
        //if the first option is chosen, assume that is not a real
        //choice, simply a default
        if (textfield.selectedIndex == 0)
        {
            if (alert_show)
            {
                alert(XD_gsValidateField + textfield.name +
XD_gsValidateRequired);
            }
            blurred.focus();
            blurred.select();
            required_elements[num] = false;
        } //end if selectedIndex empty
        else if (textfield.selectedIndex > 0)
        {
            blurred = "";
            required_elements[num] = true;
            in_required = false;
        } //end else
    }
}

```

```

    } //end select-one

    if (textfield.type == "text" || textfield.type == "textarea" ||
    textfield.type == "password")
    {
        if(textfield.value.length==0)
        {
            if (alert_show)
            {
                alert(XD_gsValidateField + blurred.name +
XD_gsValidateRequired);
            } //end alert_show
            blurred.focus();
            blurred.select();
            required_elements[num] = false;
        } //end if length empty
        else if (textfield.value.length > 0)
        {
            blurred = "";
            required_elements[num] = true;
            in_required = false;
        } //end else
    } //end if text
}

```

```

function validateRequiredElements() {
    var length = required_elements.length;
    for (var i = 0; i < length; i++){
        if (!required_elements[i]){
            if (required_elements_names[i] == "") {
                alert(XD_gsValidateAllRequiredField + i +
XD_gsValidateNotFilled);
                return false;
            } else {
                alert(required_elements_names[i] + XD_gsValidateNotFilled);
                return false;
            }
        } // end of false element
    } // end of array
    return true;
}

```

```

function validatePhone(textfield) {
    phoneOK=true;
    var digits = 0;

    //Number can only contains ten digits and proper characters
    for(var i = 0; i < textfield.value.length; i++) {
        var theChar = textfield.value.charAt(i);
        if ((theChar >= "0") && (theChar <= "9")) {
            digits++;
            continue;
        }

        if (theChar == " ") continue;
        if (theChar == "-") continue;
        if (theChar == "(") continue;
        if (theChar == ")") continue;

        //else
    }
}

```

```

    phoneOK = false;
} //end for

phoneOK = phoneOK && (digits == 10);
if (textfield.value == "") {
    phoneOK = true;
}
if (!phoneOK) {
    alert(XD_gsValidatePhoneFormat);
}

return phoneOK;
}

//Check that the date is in the form of DD-MON-YY
function validateDate(textfield) {
    dateOK=true;
    if ((textfield.value.charAt(0) > "3") || (textfield.value.charAt(0) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(1) > "9") || (textfield.value.charAt(0) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(7) > "9") || (textfield.value.charAt(7) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(8) > "9") || (textfield.value.charAt(8) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(9) > "9") || (textfield.value.charAt(9) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(10) > "9") || (textfield.value.charAt(10)
< "0")) {
        dateOK=false;
    }
    if (textfield.value.charAt(2) != "-") {
        dateOK=false;
    }
    if (textfield.value.charAt(6) != "-") {
        dateOK=false;
    }
    var month = textfield.value.substring(3, 6);
    month = month.toUpperCase();
    if (!(month == "JAN" || month == "FEB" ||
month == "MAR" || month == "APR" ||
month == "MAY" || month == "JUN" ||
month == "JUL" || month == "AUG" ||
month == "SEP" || month == "OCT" ||
month == "NOV" || month == "DEC")) {
        dateOK= false;
    }
    if (textfield.value == "") {
        dateOK = true;
    }
    if (!dateOK) {
        alert(XD_gsValidateDateFormat);
    }
}

```

```

    }
}

//Check that the date is in the form of DD/MM/YY
function validateDate_old(textfield) {
    dateOK=true;
    if ((textfield.value.charAt(0) > "9") || (textfield.value.charAt(0) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(1) > "9") || (textfield.value.charAt(0) <
"0")) {
        dateOK=false;
    }
    if (textfield.value.charAt(2) != "/"){
        dateOK=false;
    }
    if ((textfield.value.charAt(3) > "3") || (textfield.value.charAt(3) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(4) > "9") || (textfield.value.charAt(4) <
"0")) {
        dateOK=false;
    }
    if (textfield.value.charAt(5) != "/"){
        dateOK=false;
    }
    if ((textfield.value.charAt(6) > "9") || (textfield.value.charAt(7) <
"0")) {
        dateOK=false;
    }
    if ((textfield.value.charAt(7) > "9") || (textfield.value.charAt(7) <
"0")) {
        dateOK=false;
    }
    if (textfield.value == "") {
        dateOK = true;
    }
    if (!dateOK) {
        alert(XD_gsValidateDateFormat);
    }
}

```

```

// checks to see that the textfield contains only numbers
function validateNum(textfield)
{

```

```

    all_numOK = true;

    for (var i=0; i < textfield.value.length; i++)
    {
        var theChar = textfield.value.charAt(i);
        if ((theChar < "0") || (theChar > "9"))
        {
            if (textfield.value != "-1")
            {
                all_numOK = false;
                alert(XD_gsValidateContainNums);
                break;
            } // end of if not -
        }
    }
}

```

```

        } //end if not #
    } //end for

    return all_numOK;
}

// checks to see that the textfield contains two decimal places
function validateMoney(textfield) {

    all_moneyOK = true;

    for (var i=0; i < textfield.value.length; i++) {
        var theChar = textfield.value.charAt(i);
        if ((theChar < "0") || (theChar > "9")) {
            if (theChar != ".") {
                all_moneyOK = false;
                alert(XD_gsValidateMoneyFormat);
                break;
            }
        } //end if not #
    } //end for

    return all_moneyOK;
}

function validateLength(textfield, len)
{
    lengthOK = true;

    if (textfield.value.length < len)
    {
        lengthOK = false;
        alert(textfield.name + XD_gsValidateLengthFormat + len +
XD_gsValidateChars);
    }
}

// attache to delete buttons to confirm
function confirmDelete(textfield) {
    deleteOK = confirm(textfield.value + ": Are you sure?");
}

// attache to other buttons, such as add, to allow them to submit
// after a failed delete confirm
function reset_b() {
    deleteOK = true;
    emailOK = true;
    phoneOK = true;
    dateOK = true;
    lengthOK = true;
    yearOK = true;
    all_numOK = true;
    deleteOK = true;
    var length = required_elements.length;
    for (var i=0; i < length; i++) {
        required_elements[i] = true;
    }
}

// a function to error check with
function test() {
    alert("Testing!")
}

```

```

    }

    // checks to see that the textfield contains only numbers and is
    //between 13 and 16 characters in length
    function validateLengthandInput(textfield, minLength, maxLength, dateType)
    {
        all_numOK = true;

        if ((textfield.value.length<minLength) ||
            (textfield.value.length>maxLength))
        {
            all_numOK = false;

            if (minLength == maxLength)
            {
                if (dateType == "ExpDate") {
                    alert(XD_gsValidateExpDateFormat);
                }
                else {
                    alert(XD_gsValidateDateFormat + maxLength +
                        XD_gsValidateChars);
                }
            }
            else
            {
                alert(XD_gsValidateCard + minLength + XD_gsValidateAnd +
                    maxLength + XD_gsValidateChars);
            }

            return all_numOK;
        }

        for (var i=0; i < textfield.value.length; i++)
        {
            var theChar = textfield.value.charAt(i);
            if ((theChar < "0") || (theChar > "9"))
            {
                if (textfield.value != "-1")
                {
                    all_numOK = false;
                    alert(XD_gsValidateContainNums);
                    break;
                } // end of if not -
            } //end if not #
        } //end for

        return all_numOK;
    }

    function checkRequired(form)
    {
        var complete = true;
        var length = form.elements.length;

        addRequiredElements(form);
        for (var i=0; i<length; i++)
        {
            for (var j=0; j < required_elements_names.length; j++)
            {
                if (form.elements[i].name == required_elements_names[j])
                {

```

```

        if ((form.elements[i].type == "text") ||
            (form.elements[i].type == "password") ||
            (form.elements[i].type == "textarea"))
        {
            if (form.elements[i].value == '')
            {
                complete = false;
                break;
            }
        }
        else if (form.elements[i].type == "select-one")
        {
            if (form.elements[i].selectedIndex == 0)
            {
                complete = false;
                break;
            }
        }
        else
        {
            //don't worry about radio button
        }
    }

    }
    if (!complete)
    {
        //Temp bug fix: could not read any variable from english_text.js
        (scope problem?)
        //was: alert(XD_gsValidateAllRequired)
        alert("A required field is not filled out. Please make sure
all required fields are filled out before hitting submit.");
        return false;
    }
    else
    {
        /*
        Check if we've already submitted
        */

        if (!submitted)
        {
            submitted = true;
            submitCount = 0;
            //took this line out because it was breaking IE
            // and it's not used for submitting the form anyway
            //form.submit();
            return true;
        }
    }

    submitCount += 1;

    {
        var gender = "";
        var message = "";;
    }

```



```

        if (form.gender.value == "1" || form.gender.value == "2")
        {
            var gender;
            if (form.gender.value == "1")
            {
                gender = "Dude"
            }
            else
            {
                gender = "Lady"
            }
        }

        if (submitCount == 2)
        {
            message = " Hey " + gender + " give me a second while I send
info";
        }
        if (submitCount == 3)
        {
            message = "Okay... now your just pressing too much";
        }
        if (submitCount > 1 && submitCount < 4)
        {
            alert(message);
        }
    }

    return false;
}

function CheckPassword(form)
{
    var length = form.elements.length;
    var change=1;
    //Make sure passwords match
    if (form.elements[1].value !=
        form.elements[2].value)
    {
        alert(XD_gsValidatePasswords);
        change=0;
        return false;
    }

    if (change==1)
    {
        form.submit();
    }
    return true;
}

```

//xparse.js

```

function _element()
{
  this.type = "element";
  this.name = new String();
  this.attributes = new Array();
  this.contents = new Array();
  this.uid = _Xparse_count++;
  _Xparse_index[this.uid]=this;

  // Added by Martin Hald
  this.attributes.folder = 0;
}

function _chardata()
{
  this.type = "chardata";
  this.value = new String();
}

function _pi()
{
  this.type = "pi";
  this.value = new String();
}

function _comment()
{
  this.type = "comment";
  this.value = new String();
}

// an internal fragment that is passed between functions
function _frag()
{
  this.str = new String();
  this.ary = new Array();
  this.end = new String();
}

////////////////////
// global vars to track element UID's for the index
var _Xparse_count = 0;
var _Xparse_index = new Array();

////////////////////
//// Main public function that is called to
//// parse the XML string and return a root element object

function Xparse(src)
{
  // Hack added by Martin Hald to fix the grove[x] not an object error
  // where the grove object array indexes was shifted up by the previos
  // parsing
  _Xparse_count = 0;

  var frag = new _frag();

```

```

// remove bad \r characters and the prolog
frag.str = _prolog(src);

// create a root element to contain the document
var root = new _element();
root.name= XD_gsRootPath;
root.attributes.folder = 1;
root.attributes.show = 1;

// main recursive function to process the xml
frag = _compile(frag);

// all done, lets return the root element + index + document
root.contents = frag.ary;
root.index = _Xparse_index;
_Xparse_index = new Array();

return root;
}

////////////////////////////////////

////////////////////////////////////
//// transforms raw text input into a multilevel array
function _compile(frag)
{
    // keep circling and eating the str
    while (1)
    {
        // when the str is empty, return the fragment
        if (frag.str.length == 0)
        {
            return frag;
        }

        var TagStart = frag.str.indexOf("<");

        if (TagStart != 0)
        {
            // theres a chunk of characters here, store it and go on
            var thisary = frag.ary.length;
            frag.ary[thisary] = new _chardata();
            if (TagStart == -1)
            {
                frag.ary[thisary].value = _entity(frag.str);
                frag.str = "";
            }
            else
            {
                frag.ary[thisary].value =
                _entity(frag.str.substring(0,TagStart));
                frag.str = frag.str.substring(TagStart,frag.str.length);
            }
        }
        else
        {
            // determine what the next section is, and process it
            if (frag.str.substring(1,2) == "?")
            {
                frag = _tag_pi(frag);
            }
            else

```

```

    {
      if (frag.str.substring(1,4) == "!--")
      {
        frag = _tag_comment(frag);
      }
      else
      {
        if (frag.str.substring(1,9) == "![CDATA[")
        {
          frag = _tag_cdata(frag);
        }
        else
        {
          if (frag.str.substring(1,frag.end.length + 3) == "/"
+ frag.end + ">" || _remove_escapes(frag.str.substring(1,frag.end.length +
3)) == "/" + frag.end)
          {
            // found the end of the current tag, end the
recursive process and return
            frag.str = frag.str.substring(frag.end.length +
3,frag.str.length);
            frag.end = "";
            return frag;
          }
          else
          {
            frag = _tag_element(frag);
          }
        }
      }
    }
  }
}
return "";
}

```

```

////////////////////////////////////

```

```

////////////////////////////////////
//// functions to process different tags
function XDTrueSpaceIndex(frag)
{
  var length = frag.length;
  for (var i=0; i < length; i++)
  {
    if ( (frag.charAt(i) == " ")
      &&(frag.charAt(i-1) != "\\")
    )
    {
      break;
    }
  }
  return i;
}

```

```

function _tag_element(frag)
{

```

```

// initialize some temporary variables for manipulating the tag
var close = frag.str.indexOf(">");
var empty = (frag.str.substring(close - 1,close) == "/");
if (empty)
{
    close -= 1;
}

// split up the name and attributes
var starttag = _normalize(frag.str.substring(1,close));
//var nextspace = starttag.indexOf(" ");
var nextspace = XDTrueSpaceIndex(starttag);
var attribs = new String();
var name = new String();
if (nextspace != -1)
{
    name = starttag.substring(0,nextspace);
    attribs = starttag.substring(nextspace + 1,starttag.length);
}
else
{
    name = starttag;
}

var thisary = frag.ary.length;
frag.ary[thisary] = new _element();
frag.ary[thisary].name = _remove_escapes(name);

if (attribs.length > 0)
{
    frag.ary[thisary].attributes = _attribution(attribs);
}

if (!empty)
{
    // !!!! important,
    // take the contents of the tag and parse them
    var contents = new _frag();
    contents.str = frag.str.substring(close + 1,frag.str.length);
    contents.end = name;
    contents = _compile(contents);
    frag.ary[thisary].contents = contents.ary;
    frag.str = contents.str;
}
else
{
    frag.str = frag.str.substring(close + 2,frag.str.length);
}
return frag;
}

function _tag_pi(frag)
{
    var close = frag.str.indexOf(">");
    var val = frag.str.substring(2,close);
    var thisary = frag.ary.length;
    frag.ary[thisary] = new _pi();
    frag.ary[thisary].value = val;
    frag.str = frag.str.substring(close + 2,frag.str.length);
    return frag;
}

function _tag_comment(frag)

```

```

{
  var close = frag.str.indexOf("-->");
  var val = frag.str.substring(4,close);
  var thisary = frag.ary.length;
  frag.ary[thisary] = new _comment();
  frag.ary[thisary].value = val;
  frag.str = frag.str.substring(close + 3,frag.str.length);
  return frag;
}

```

```

function _tag_cdata(frag)
{
  var close = frag.str.indexOf("]]>");
  var val = frag.str.substring(9,close);
  var thisary = frag.ary.length;
  frag.ary[thisary] = new _chardata();
  frag.ary[thisary].value = val;
  frag.str = frag.str.substring(close + 3,frag.str.length);
  return frag;
}

```

```

////////////////////

```

```

////////////////////
//// util for element attribute parsing
//// returns an array of all of the keys = values
function _attribution(str)

```

```

{
  var all = new Array();
  while (1)
  {
    var eq = str.indexOf("=");
    if (str.length == 0 || eq == -1)
    {
      return all;
    }

    var id1 = str.indexOf("'");
    var id2 = str.indexOf("\"");
    var ids = new Number();
    var id = new String();
    if ((id1 < id2 && id1 != -1) || id2 == -1)
    {
      ids = id1;
      id = "'";
    }
    if ((id2 < id1 || id1 == -1) && id2 != -1)
    {
      ids = id2;
      id = "\"";
    }
    var nextid = str.indexOf(id,ids + 1);
    var val = str.substring(ids + 1,nextid);

    var name = xstrip(str.substring(0,eq));
    var entity = new String();
    entity = _entity(val);
    all[name] = entity;
    str = str.substring(nextid + 1,str.length);
  }
  return all;
}

```

```

////////////////////

////////////////////
//// util to remove \r characters from input string
//// and return xml string without a prolog
function _prolog(str)
{
    var A = new Array();

    A = str.split("\r\n");
    str = A.join("\n");
    A = str.split("\r");
    str = A.join("\n");

    var start = str.indexOf("<");
    if (str.substring(start,start + 3) == "<?x" || str.substring(start,start
+ 3) == "<?X" )
    {
        var close = str.indexOf(">");
        str = str.substring(close + 2,str.length);
    }
    var start = str.indexOf("<!DOCTYPE");
    if (start != -1)
    {
        var close = str.indexOf(">",start) + 1;
        var dp = str.indexOf("[",start);
        if (dp < close && dp != -1)
        {
            close = str.indexOf("]>",start) + 2;
        }
        str = str.substring(close,str.length);
    }
    return str;
}
////////////////////

function _remove_escapes (str)
{
    var A = new Array();
    A = str.split("\\");
    str = A.join("");
    return str;
}

////////////////////
//// util to remove white characters from input string
function xstrip(str)
{
    A = str.split(" ");
    str = A.join("");
    A = str.split("\n");
    str = A.join("");
    A = str.split("\t");
    str = A.join("");

    //A = str.split(" ");
    //str = A.join("&nbsp;");
    //A = str.split("\n");
    //str = A.join("");
    //A = str.split(" ");
    //str = A.join("");

```

```
//A = str.split("\t");
//str = A.join("");

return str;
}
//////////

//////////
//// util to replace white characters in input string
function _normalize(str)
{
    var A = new Array();

    A = str.split("\n");
    str = A.join(" ");
    A = str.split("\t");
    str = A.join(" ");

    return str;
}
//////////

//////////
//// util to replace internal entities in input string
function _entity(str)
{
    var A = new Array();

    //A = str.split("<");
    //str = A.join("<");
    //A = str.split(">");
    //str = A.join(">");
    //A = str.split(""");
    //str = A.join(""");
    //A = str.split("&apos;");
    //str = A.join("&apos;");

    //A = str.split("&");
    //str = A.join("&");

    //Get rid of any escapes
    A = str.split("\\");
    str = A.join("");

    return str;
}
//////////
```


CLAIMS

What is claimed is:

1. A shared computer network storage system, comprising:
 - a first database containing file data;
 - a second database containing information (metadata) about said file data of said first database;
 - a server, said server executing file commands on said first file database, said server contemporaneously updating said second metadatabase upon executing said file commands; and
 - a client application, said client application communicating with said server, said client application invoking file commands upon said server, said server executing said file commands and updating information regarding said first file and second metadata databases displayed by said client application; wherebysaid client application controls files in said first file database and information regarding status of said first database files is more readily available by reference to said second metadatabase.
2. The shared computer network storage system of claim 1, wherein said first file database is distributed over at least two physical storage devices.
3. The shared computer network storage system of claim 1, wherein said second metadatabase is distributed over at least two physical storage devices.
4. The shared computer network storage system of claim 1, wherein said client application communicates with said server via a proxy.
5. The shared computer network storage system of claim 1, wherein said server comprises a non-routable network.
6. The shared computer network storage system of claim 1, wherein said server comprises a transaction processor.
7. The shared computer network storage system of claim 6, wherein said transaction processor guarantees access to and transactions on said first and second databases.
8. The shared computer network storage system of claim 1, wherein said server comprises an enterprise java bean cluster (EJBC).
9. The shared computer network storage system of claim 8, wherein said enterprise java bean cluster (EJBC) handles business logic and resource access methods as well as memory caching for common resources.
10. The shared computer network storage system of claim 1, wherein said server further comprises an application network.

11. The shared computer network storage system of claim 10, wherein said application network further
2 comprises a java application cluster.
12. The shared computer network storage system of claim 10, wherein said application network handles
2 display functions and resource requests.
13. The shared computer network storage system of claim 1, wherein said server further comprises a web
2 server.
14. The shared computer network storage system of claim 13, wherein said web server handles all requests
2 for static content and proxies requests for dynamic content.
15. The shared computer network storage system of claim 1, wherein said server further comprises a load
2 balancer, said load balancer proxying requests to a sub-server having the highest degree of availability or
functionality.
16. The shared computer network storage system of claim 1 wherein said server further comprises a DNS
2 redirector, said DNS redirector proxying requests to a resource having a highest degree of functionality.
17. The shared computer network storage system of claim 1 wherein said server further comprises:
2 a transaction processor, said transaction processor on a non-routable network, said transaction
processor guarantees access to and transactions on said first and second databases;
4 an enterprise java bean cluster (EJBC) on a non-routable network, said enterprise java bean cluster
(EJBC) coupled to said transaction processor and handling business logic and resource access methods a
6 well as memory caching for common resources;
an application network on a non-routable network, said application network coupled to said
8 enterprise java bean cluster, said application network including a java application cluster and handling
display functions and resource requests;
10 a web server, said web server coupled to said application network and handling all requests for
static content and proxies requests for dynamic content;
12 a load balancer, said load balancer coupled to said web server and proxying requests to a sub-
server having the highest degree of availability or functionality; and
14 a DNS redirector, said DNS redirector coupled to said load balancer and proxying requests to a
resource having a highest degree of functionality.
18. The shared computer network storage system of claim 1, wherein said client application is web-based.
19. The shared computer network storage system of claim 1, wherein said client application interacts with
2 an operating system running upon a computer upon which said client application is also running, said client
application adopting and implementing a visual display format similar to said operating system.

20.

A shared computer network storage system, comprising:

a first database containing file data, said first database distributed over at least two physical storage devices;

a second database containing information (metadata) about said file data of said first database, said second database distributed over at least two physical storage devices;

a server, said server executing file commands on said first file database, said server contemporaneously updating said second metadatabase upon executing said file commands, said server including:

a transaction processor, said transaction processor on a non-routable network, said transaction processor guarantees access to and transactions on said first and second databases;

an enterprise java bean cluster (EJBC) on a non-routable network, said enterprise java bean cluster (EJBC) coupled to said transaction processor and handling business logic and resource access methods as well as memory caching for common resources;

an application network on a non-routable network, said application network coupled to said enterprise java bean cluster, said application network including a java application cluster and handling display functions and resource requests;

a web server, said web server coupled to said application network and handling all requests for static content and proxies requests for dynamic content;

a load balancer, said load balancer coupled to said web server and proxying requests to a sub-server having the highest degree of availability or functionality; and

a DNS redirector, said DNS redirector coupled to said load balancer and proxying requests to a resource having a highest degree of functionality; and

a client application, said client application communicating with said server via a proxy, said client application invoking file commands upon said server, said server executing said file commands and updating information regarding said first file and second metadata databases displayed by said client application; whereby

said client application controls files in said first file database and information regarding status of said first database files is more readily available by reference to said second metadatabase.

21.

The shared computer network storage system of claim 20, wherein said client application is web-based.

22.

The shared computer network storage system of claim 20, wherein said client application interacts with an operating system running upon a computer upon which said client application is also running, said client application adopting and implementing a visual display format similar to said operating system.

23.

A method for providing private file space and information transfer over a public computer network, the steps comprising:

providing a publicly-available private file space system coupled to the public computer network;

providing a client program in communication with the public computer network;

sending a request from said client program to said publicly-available private file space system

6 ("private system");
evaluating said request;
8 authenticating said request;
satisfying said request; and
10 returning a success indicator to said client program indicating the success or failure of said request;
whereby
12 said client program may create and control files held by said private system.

24. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 23, wherein the step of evaluating said request further comprises evaluating said request for
static content and returning an appropriate response if said request is for static content.

25. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 24, the steps further comprising:
providing an application network within said private system;
4 proxying said request to said application network; and
parsing a header of said request.

26. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 23, wherein said step of authenticating said request further comprises:
authenticating a user using said client program; and
4 authenticating said request made by said client program to ensure that it conforms with an account
associated with said user.

27. The method for providing private file space and information transfer over a public computer network
2 as set forth in claim 23, further comprising:
parsing multipart form data associated with said request;
4 determining said request's type; and
submitting said request.

28. A method for providing private file space and information transfer over a public computer network,
2 the steps comprising:
providing a publicly-available private file space system coupled to the public computer network;
4 providing a client program in communication with the public computer network;
sending a request from said client program to said publicly-available private file space system
6 ("private system");
evaluating said request for static content and returning an appropriate response if said request is for
8 static content;
providing an application network within said private system;
10 proxying said request to said application network; and
parsing a header of said request

12 authenticating said request by authenticating a user using said client program and authenticating
said request made by said client program to ensure that it conforms with an account associated with said
14 user;

parsing multipart form data associated with said request;

16 determining said request's type;

submitting said request;

18 satisfying said request; and

returning a success indicator to said client program indicating the success or failure of said request;

20 whereby

said client program may create and control files held by said private system.

29. A data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network, comprising:

a user data object;

4 a process request object; and

a recovery object;

6 said user information object, said process request object, and said recovery object associated within
a file action object.

30. The data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network as set forth in claim 29, wherein said user data object further comprises:

4 a user information object; and

a security object.

31. The data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network as set forth in claim 29, wherein said process request object further comprises:
a file operation object comprising said recovery object and a database IO object, a file IO object,
4 and an administration object.

32. The data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network as set forth in claim 29, wherein said recovery object further comprises:

a recovery IO object;

4 a mount status object;

a recovery administration object; and

6 a recovery process object.

33. A data structure for effecting file operations on a private file space and information transfer system
2 over a public computer network, comprising:

a user data object, said user data object having a user information object; and a security object;

4 a process request object, said process request object including a file operation object, a database IO
object, a file IO object, and an administration object; and

a recovery object, said recovery object incorporated in said file operation object, said recovery object including a recovery IO object, a mount status object, a recovery administration object, and a recovery process object;

said user information object, said process request object, and said recovery object associated within a file action object; whereby

file operations may be facilitated by the data structure including recovery from resource failure.

34. A shared file storage resource for a computer network, comprising:

an allocatable file storage resource;

a server, said server coupled to said storage resource, said server:

allocating individual user file space for a plurality of users on said storage resource;

receiving files for storage on said storage resource;

transmitting files stored on said storage resource;

generating control-protocol codes for transmitting said files;

receiving file commands for controlling files on said storage resource; and

transmitting display codes indicating file status on said storage resource, said display codes representing said storage resource as a network drive;

a first network connection, said first network connection coupling said server to the computer network;

a workstation, said workstation:

receiving files for storage on said storage resource;

transmitting files stored on said storage resource;

receiving file commands for controlling files on said storage resource; and

transmitting display codes indicating file status on said storage resource, said display codes representing said storage resource as a network drive; whereby

a user may store, retrieve, and control files in a unique and secure file storage area on said allocatable storage resource available throughout the computer network and detached from said workstation.

35. The shared file storage resource for a computer network as set forth in claim 34, wherein said display codes further comprise:

a browser-interpretable object, such as a JavaScript object, said object displaying file status on said storage resource as a web page.

36. The shared file storage resource for a computer network as set forth in claim 34, further comprising:

a standalone program running on said workstation, said standalone program interpreting said display codes and providing a seamless interface to said user, said seamless interface presenting said storage resource as a local or network resource to said user and allowing said user to manipulate files on said storage resource in the same manner as local storage resources such as a floppy disk drive or a local hard drive.

37. The shared file storage resource for a computer network as set forth in claim 34, wherein said computer network, further comprises:
the Internet.

38. A method for transferring data from a first network resource to a second network resource at the direction of a user, the steps comprising:
submitting a first file location indicating data to be transferred to the second network resource;
the second network resource requesting said data at said first file location from the first network resource;
the first network resource transmitting said data to the second network resource; and
the second network resource notifying the user of successful transfer upon successful reception of said data; whereby
the user may use the first and second network resources to obtain and control said data.

39. The method for transferring data as set forth in claim 38, wherein the second network resource comprises a subscriber-based system of network-available storage space.

40. The method for transferring data as set forth in claim 38, wherein the first and second network resources are coupled to the Internet.

41. The method for transferring data as set forth in claim 38, the steps further comprising:
displaying to the user a status of transmission of said data from said first network resource to said second network resource.

42. The method for transferring data as set forth in claim 38, the steps further comprising:
verifying the user as a subscriber to or member of the second network resource.

43. A method for transferring data from a first network resource to a second network resource at the direction of a user, the steps comprising:
submitting a first file location indicating data to be transferred to the second network resource, the second network resource being a subscriber-based system of network-available data storage space;
verifying the user as a subscriber to or member of the second network resource;
the second network resource requesting said data at said first file location from the first network resource;
the first network resource transmitting said data to the second network resource via Internet;
displaying to the user a status of transmission of said data from said first network resource to said second network resource; and
the second network resource notifying the user of successful transfer upon successful reception of said data; whereby
the user may use the first and second network resources to obtain and control said data.

44. A client-server system for a network-based data storage and manipulation system, comprising:
a client system, said client system having a file access service and a file manipulation service;
a server, said server providing network-based data storage resources and responding to requests transmitted by said client system, said server effecting said requests;
said server determining if a client request is one for metadata regarding data stored upon said server;
said server providing said metadata if said client request is for metadata and transmitting said metadata to said file manipulation service; and
said server performing a file action if said client request is not for metadata, said server updating said metadata and transmitting said metadata to said file manipulation service; whereby
said server operates, and said client system presents, operations on said server in a manner similar to operations local to said client system.

45. The client-server system for a network-based data storage and manipulation system as set forth in claim 44, wherein said file access service further comprises:
a request processing layer for processing requests; and
a first network I/O layer for transmitting said requests to said server.

46. The client-server system for a network-based data storage and manipulation system as set forth in claim 44, wherein said file manipulation service further comprises:
a parser, said parser parsing said metadata from said server;
a data structure, said data structure receiving and preserving parsed data from said parser; and
a data display layer, said data display layer operating upon and displaying said parsed data;
whereby
metadata may be displayed to inform about data stored upon said server.

47. The client-server system for a network-based data storage and manipulation system as set forth in claim 46, wherein said parser is an XML parser.

48. The client-server system for a network-based data storage and manipulation system as set forth in claim 44, wherein said server further comprises:
a second network I/O layer, said second network I/O layer engaged when said requests are not for metadata, said second network I/O layer transmitting requests for file action; and
a resource access layer, said resource access layer receiving transmissions from said second network I/O layer and effecting said requests, said resource access layer engaged when said requests are for metadata, said resource access layer obtaining and transmitting said metadata; and
a metadata compiler, said metadata compiler receiving said metadata from said resource access layer, compiling said metadata, and transmitting said compiled metadata to said client system.

49. The client-server system for a network-based data storage and manipulation system as set forth in claim 48, wherein said metadata compiler is an XML generator.

50.

A client-server system for a network-based data storage and manipulation system, comprising:

a client system, said client system having a file access service and a file manipulation service;

a server, said server providing network-based data storage resources, said server creating and maintaining metadata regarding stored data, said server responding to requests transmitted by said client system, said server effecting said requests;

said server determining if a client request is one for metadata;

said server providing said metadata if said client request is for metadata and transmitting said metadata to said file manipulation service;

said server performing a file action if said client request is not for metadata, said server updating said metadata and transmitting said metadata to said file manipulation service;

said file access service having a request processing layer for processing requests and a first network I/O layer for transmitting said requests to said server;

said file manipulation service having an XML parser, said XML parser parsing said metadata from said server, said file manipulation service having a data structure, said data structure receiving and preserving parsed data from said parser, and said file manipulation service having a data display layer, said data display layer operating upon and displaying said parsed data so that metadata may be displayed to inform about data stored upon said server; and

said server having a second network I/O layer, said second network I/O layer engaged when said requests are not for metadata, said second network I/O layer transmitting requests for file action, said server having a resource access layer, said resource access layer receiving transmissions from said second network I/O layer and effecting said requests, said resource access layer engaged when said requests are for metadata, said resource access layer obtaining and transmitting said metadata, and said server having a metadata compiler in the form of an XML generator, said metadata compiler receiving said metadata from said resource access layer, compiling said metadata, and transmitting said compiled metadata to said client system; whereby

said server operates as and said client system presents operations on said server in a manner similar to operations local to said client system.

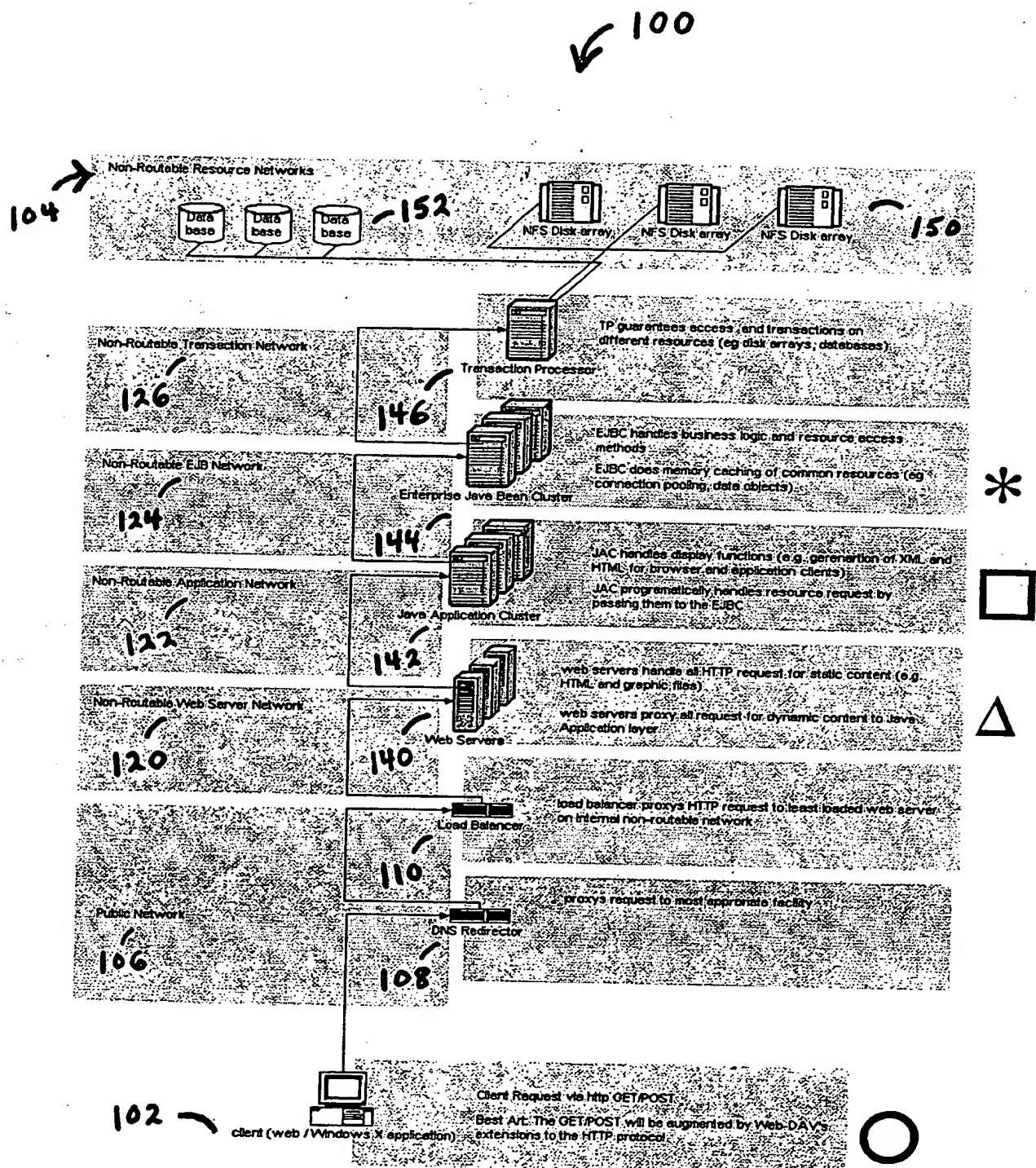


Fig. 1

Fig. 2

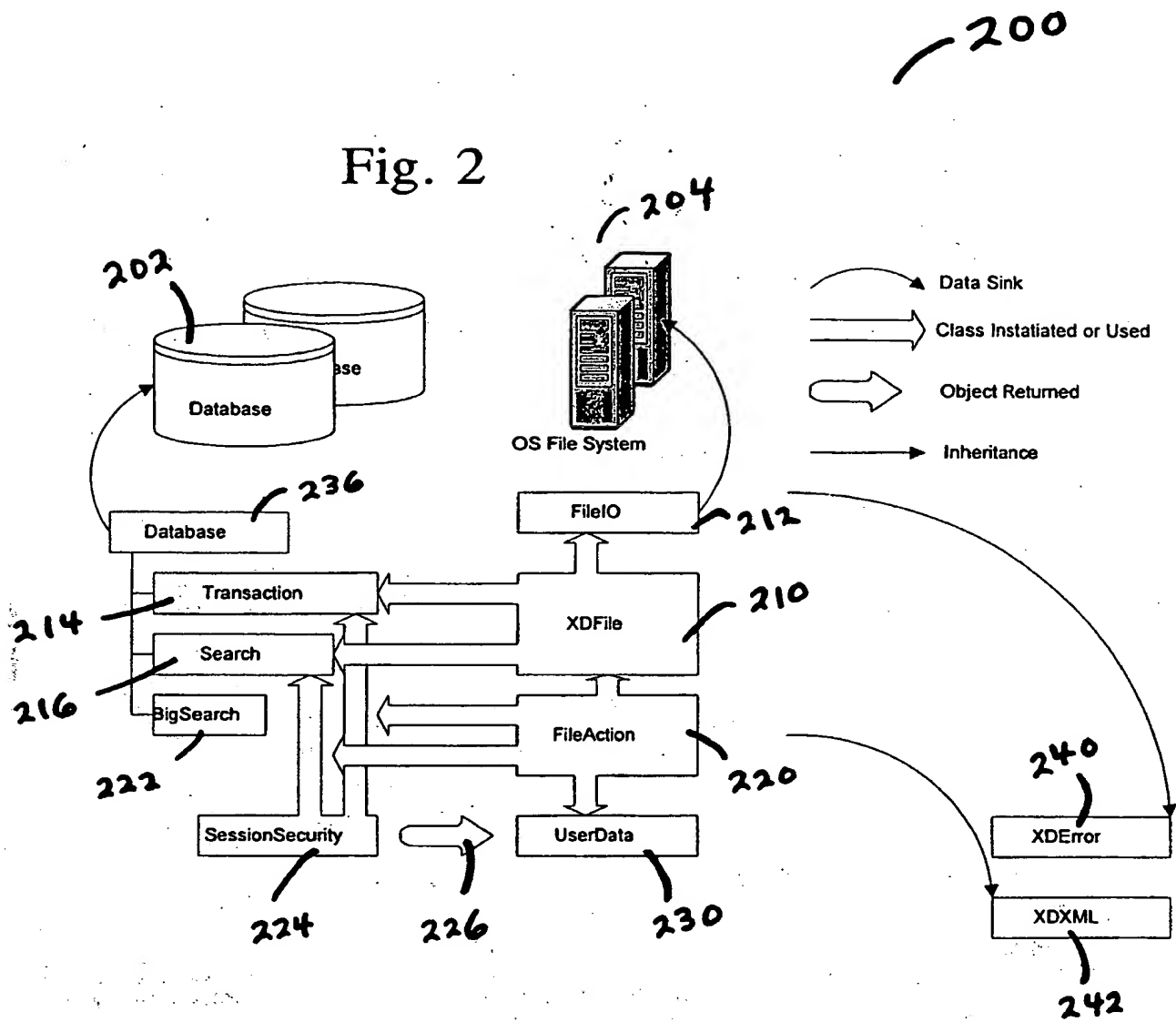


Fig. 3

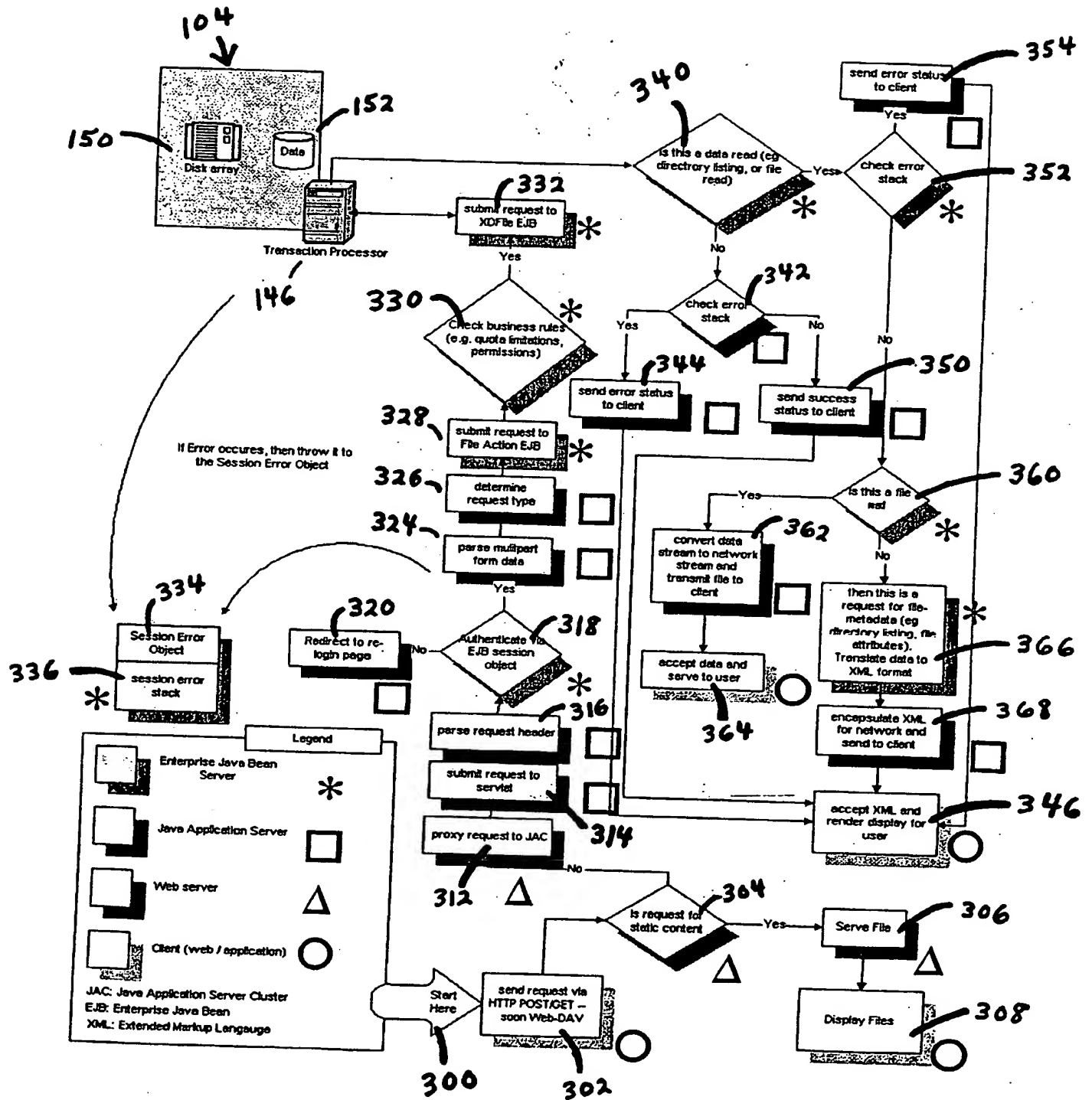
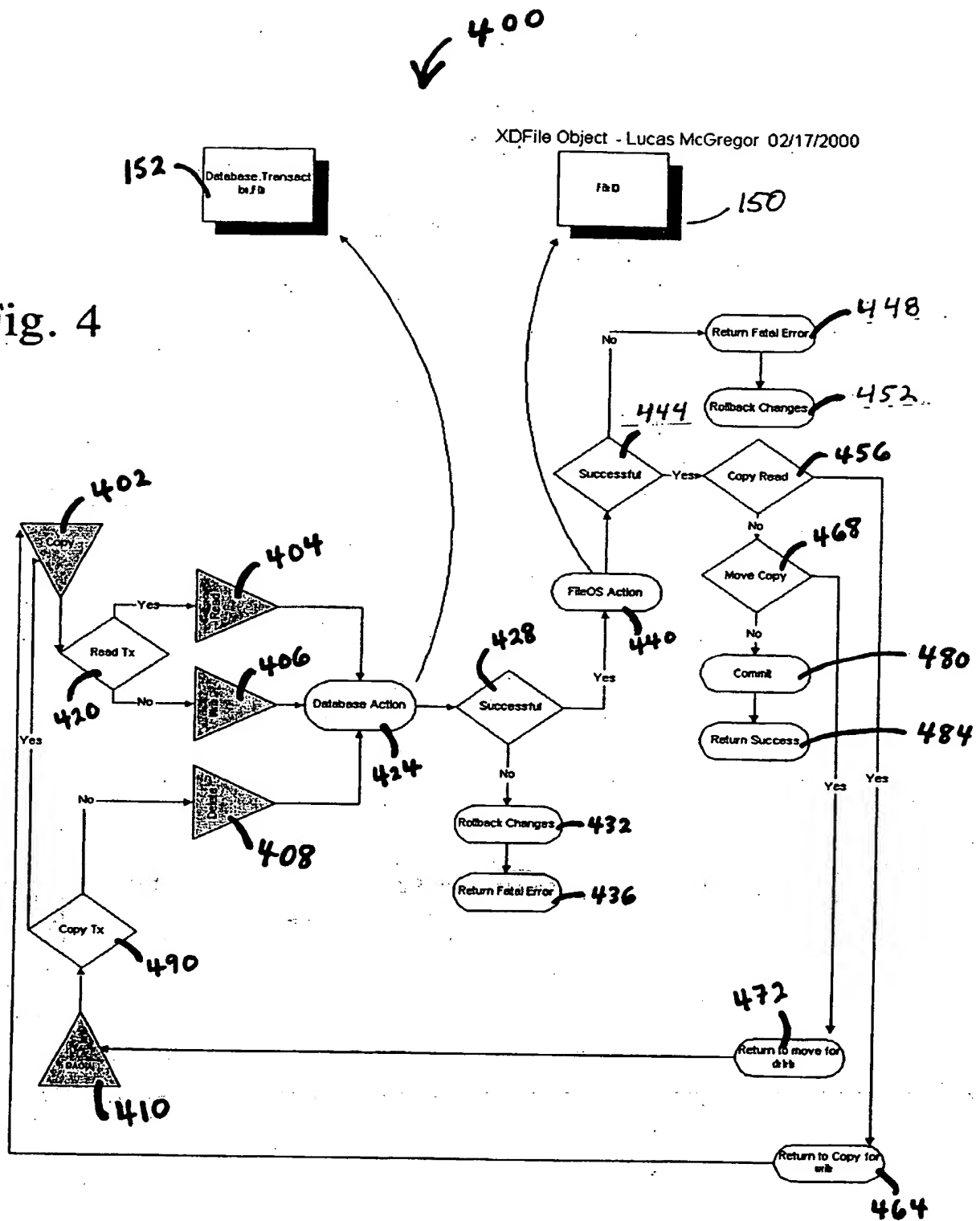
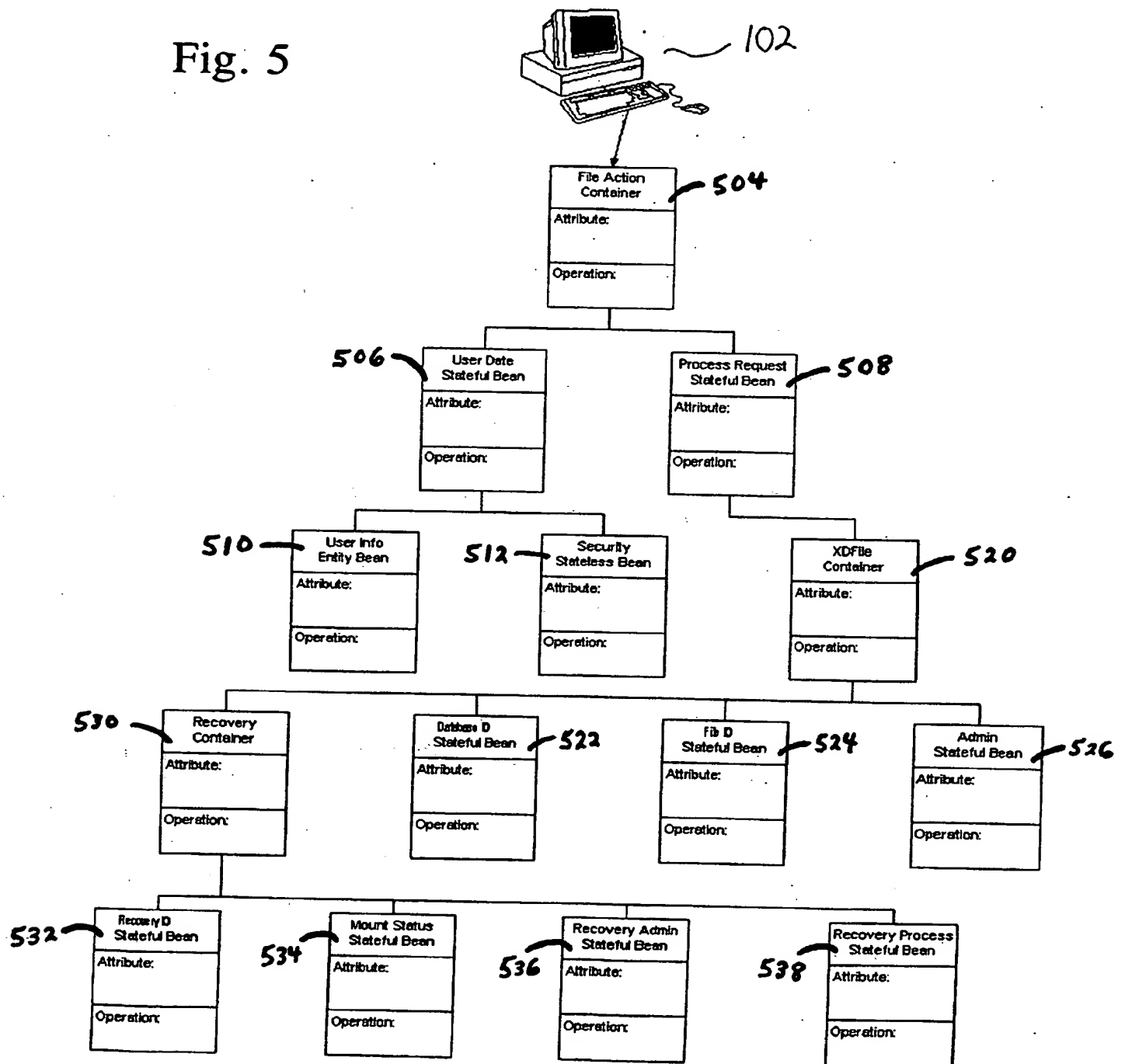


Fig. 4



TITLE: File Action, XDFile, and Recovery Object Model

Fig. 5



TITLE: File Action, XDFile, and Recovery Object Model

Fig. 6

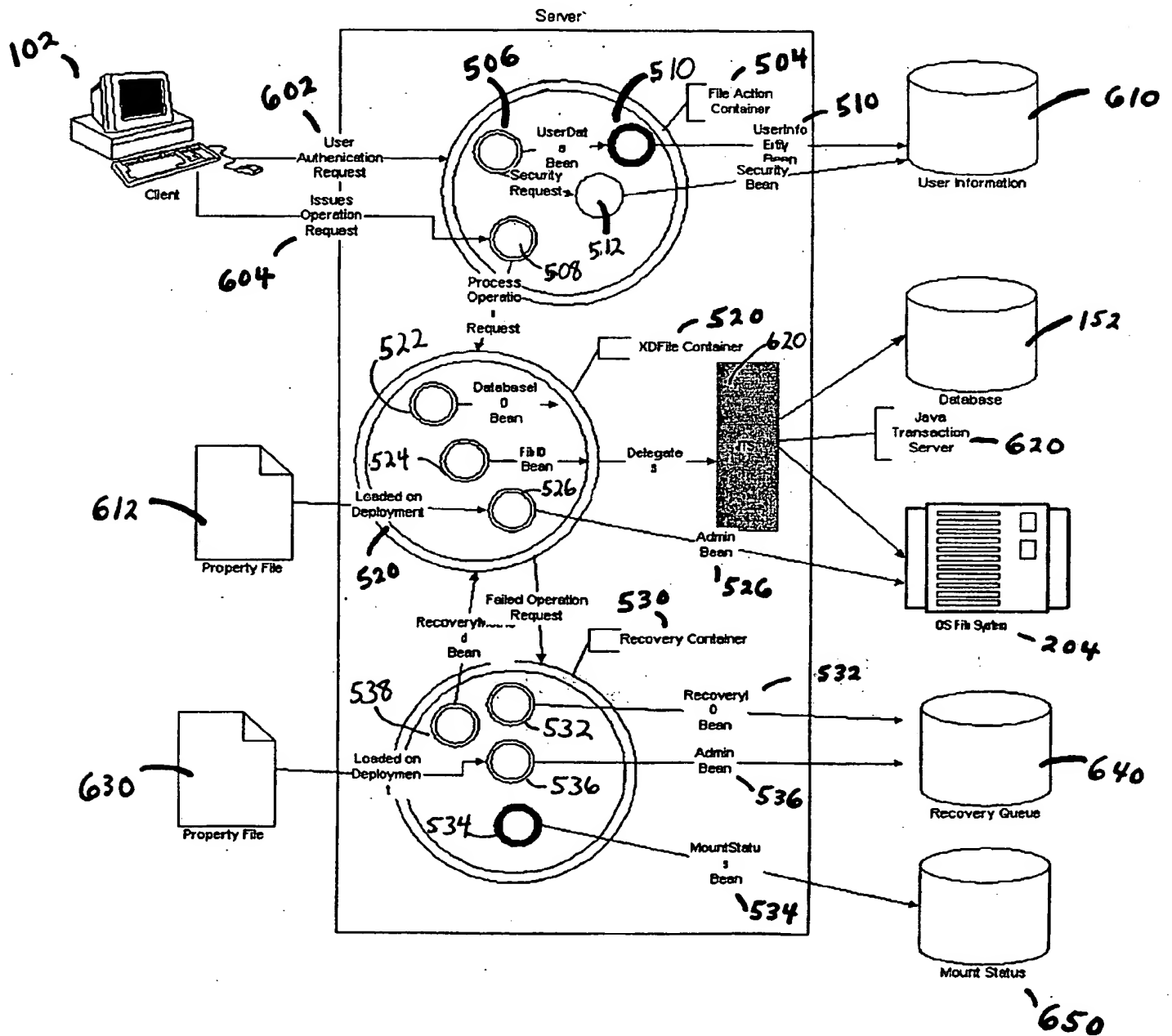


Fig. 7

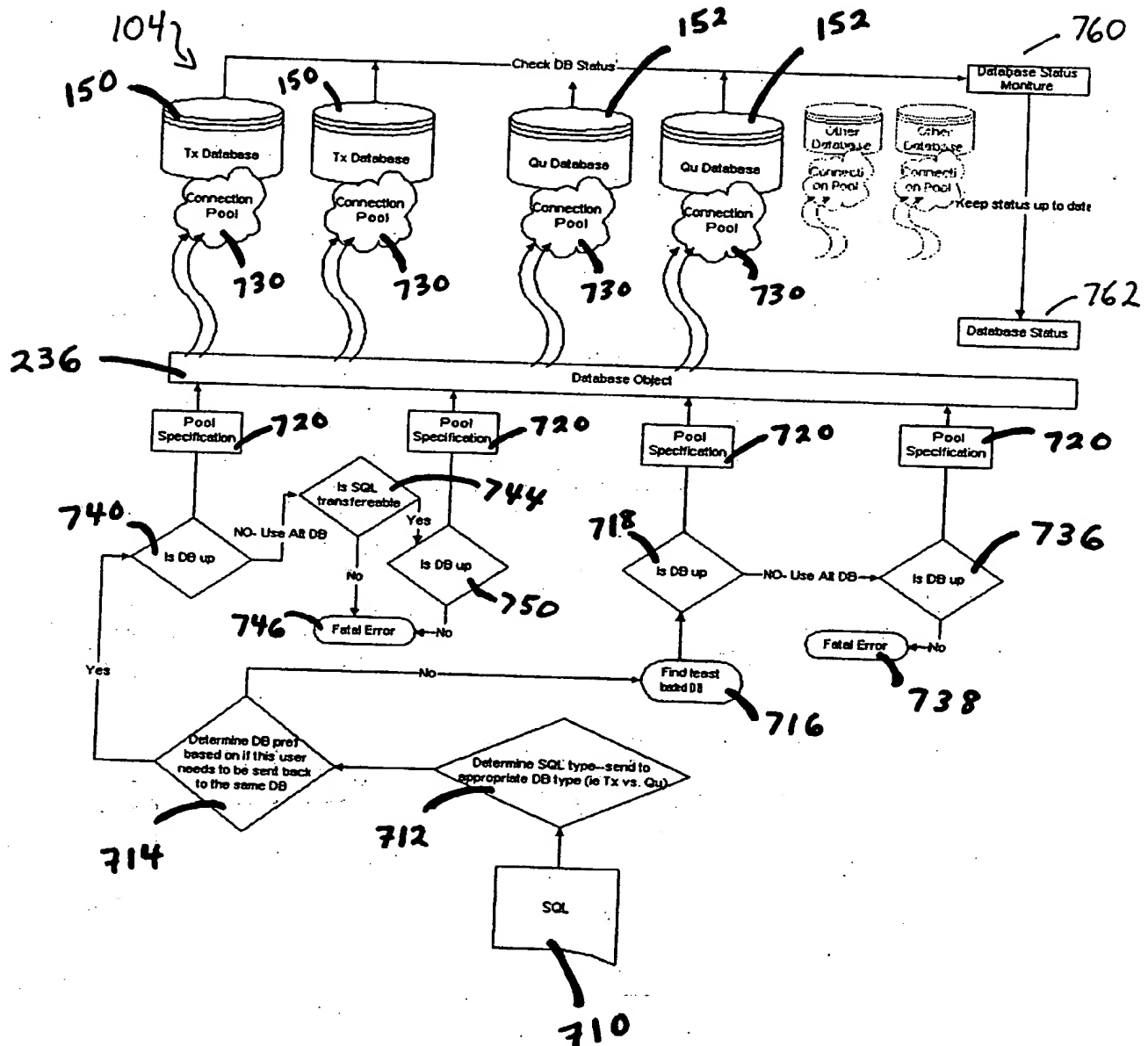


Fig. 8

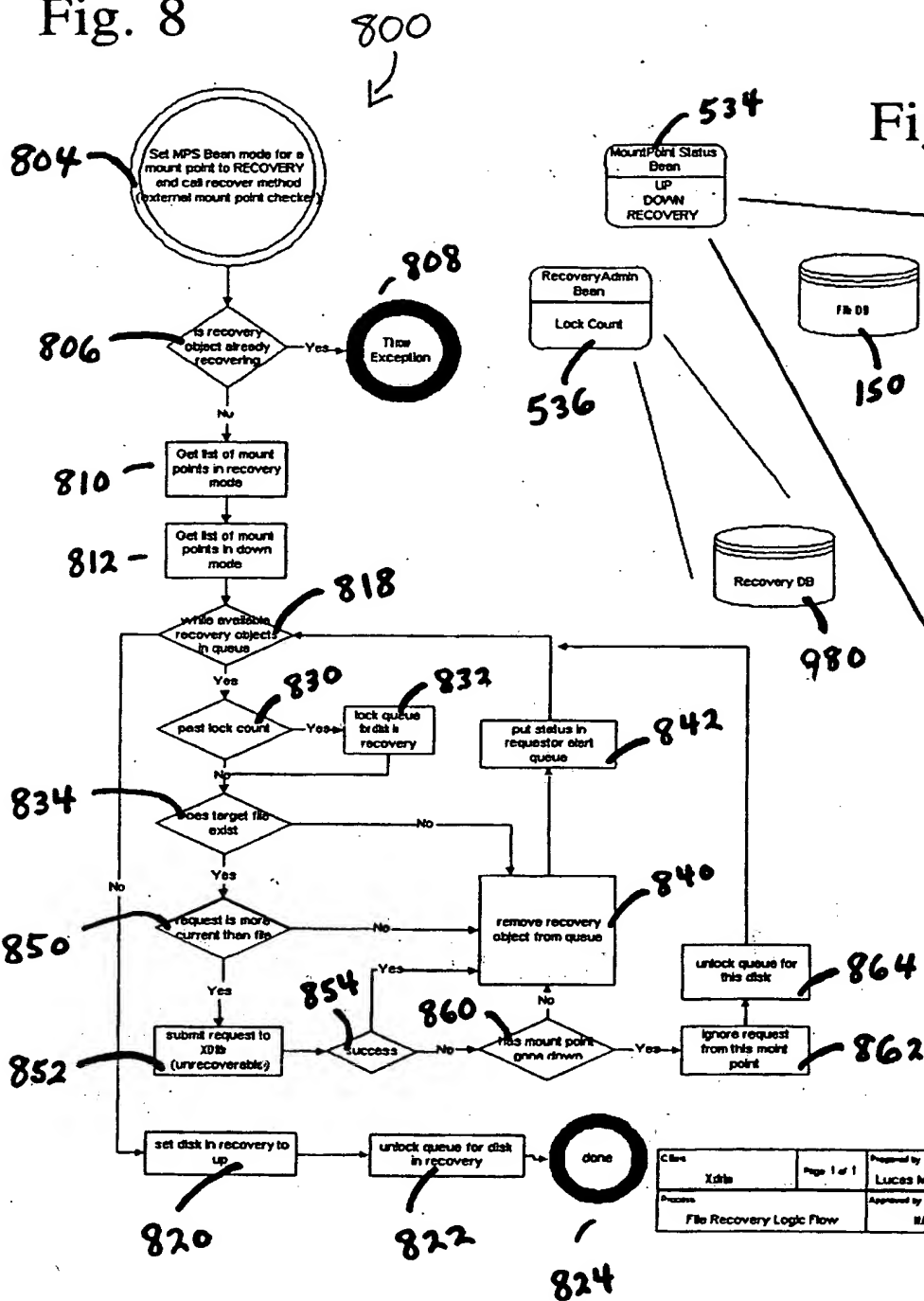
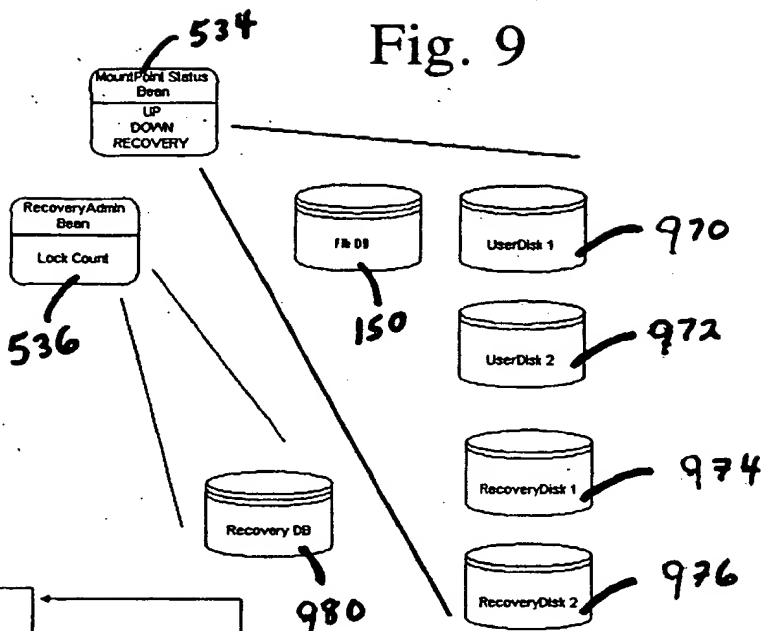


Fig. 9



Client	Idle	Page 1 of 1	Prepared by	Lucas McGregor	Date	2000
Process	File Recovery Logic Flow	Approved by	RA	Date	2000	

Fig. 10

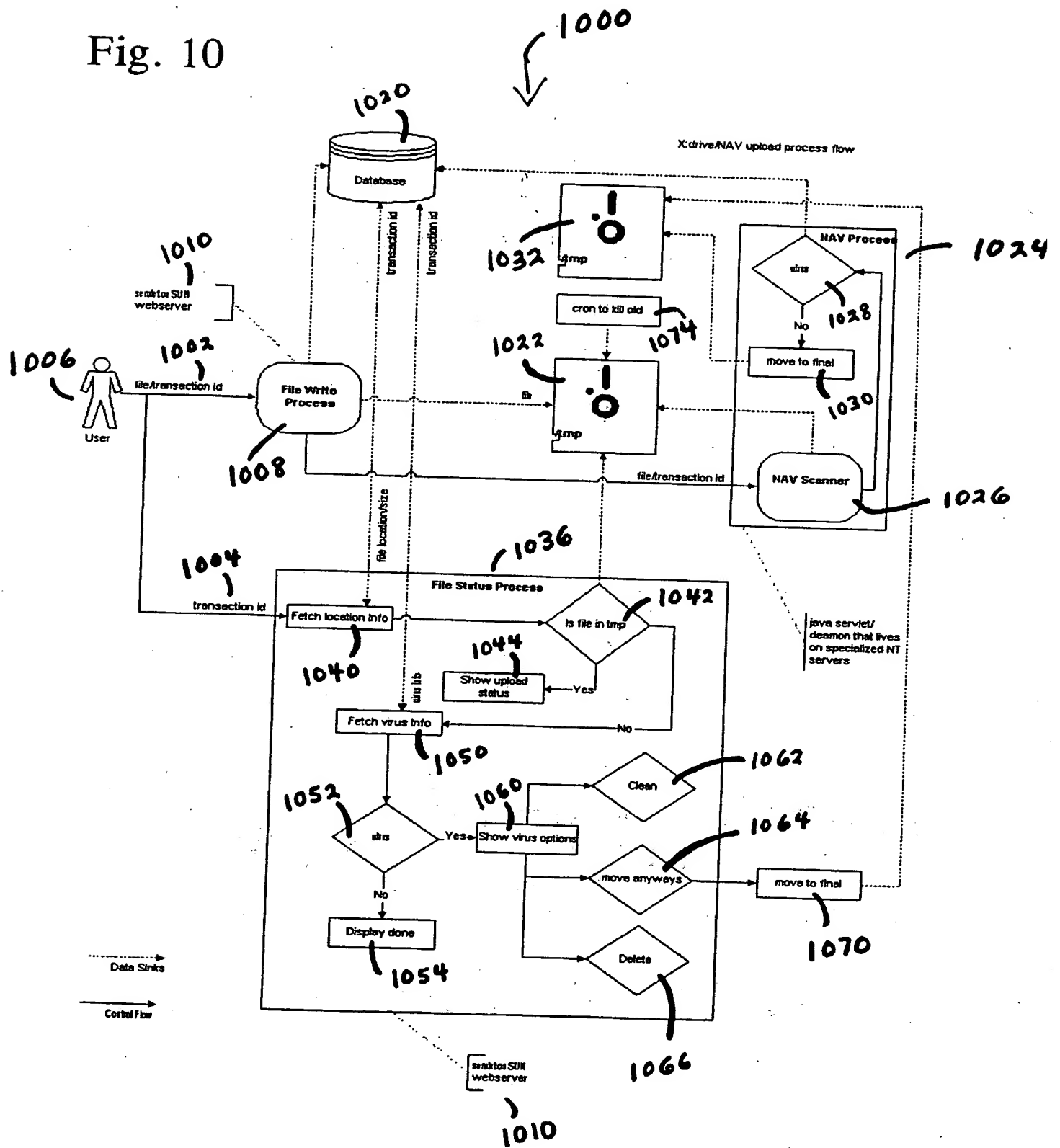


Fig. 11

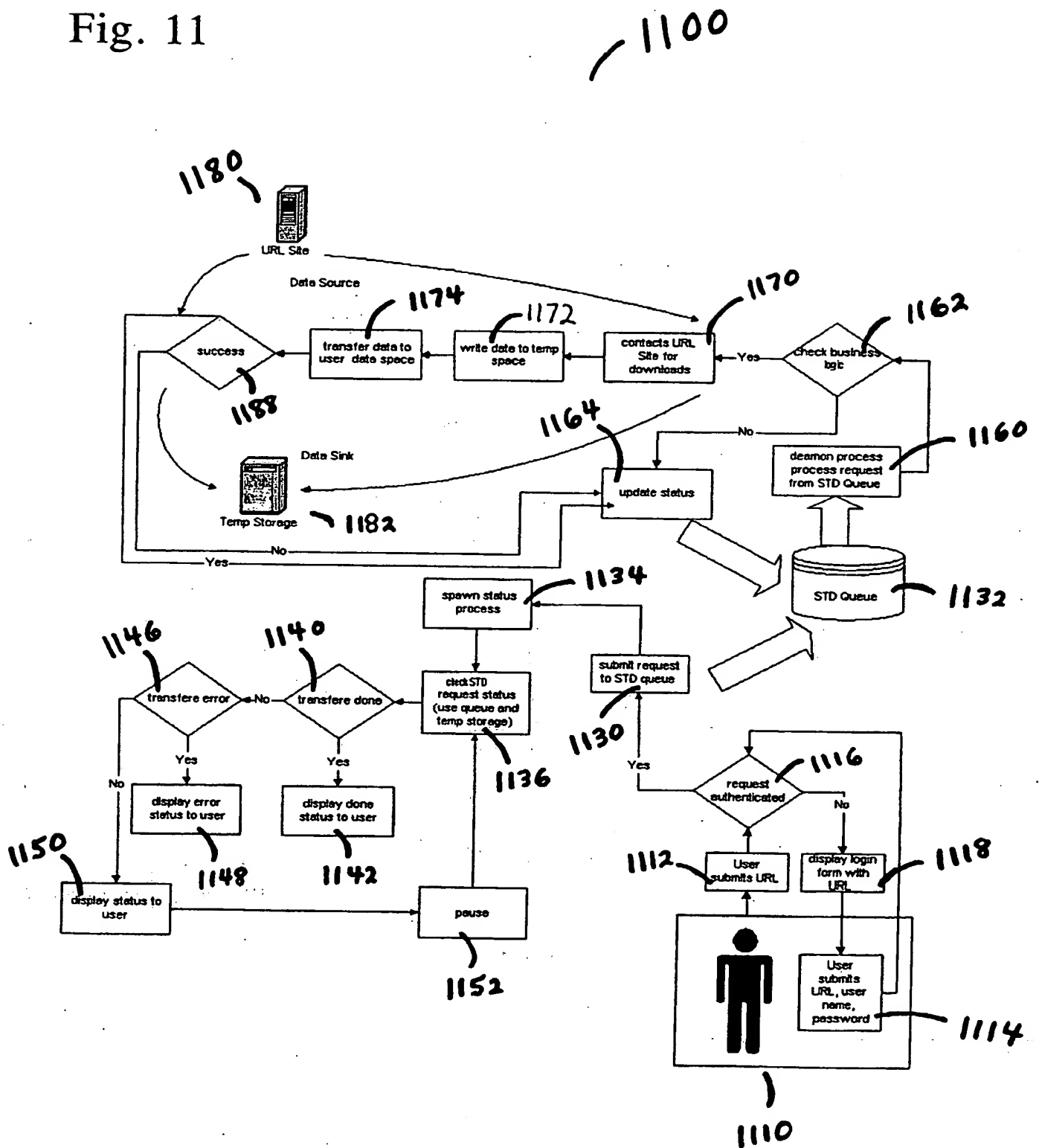


Fig. 12

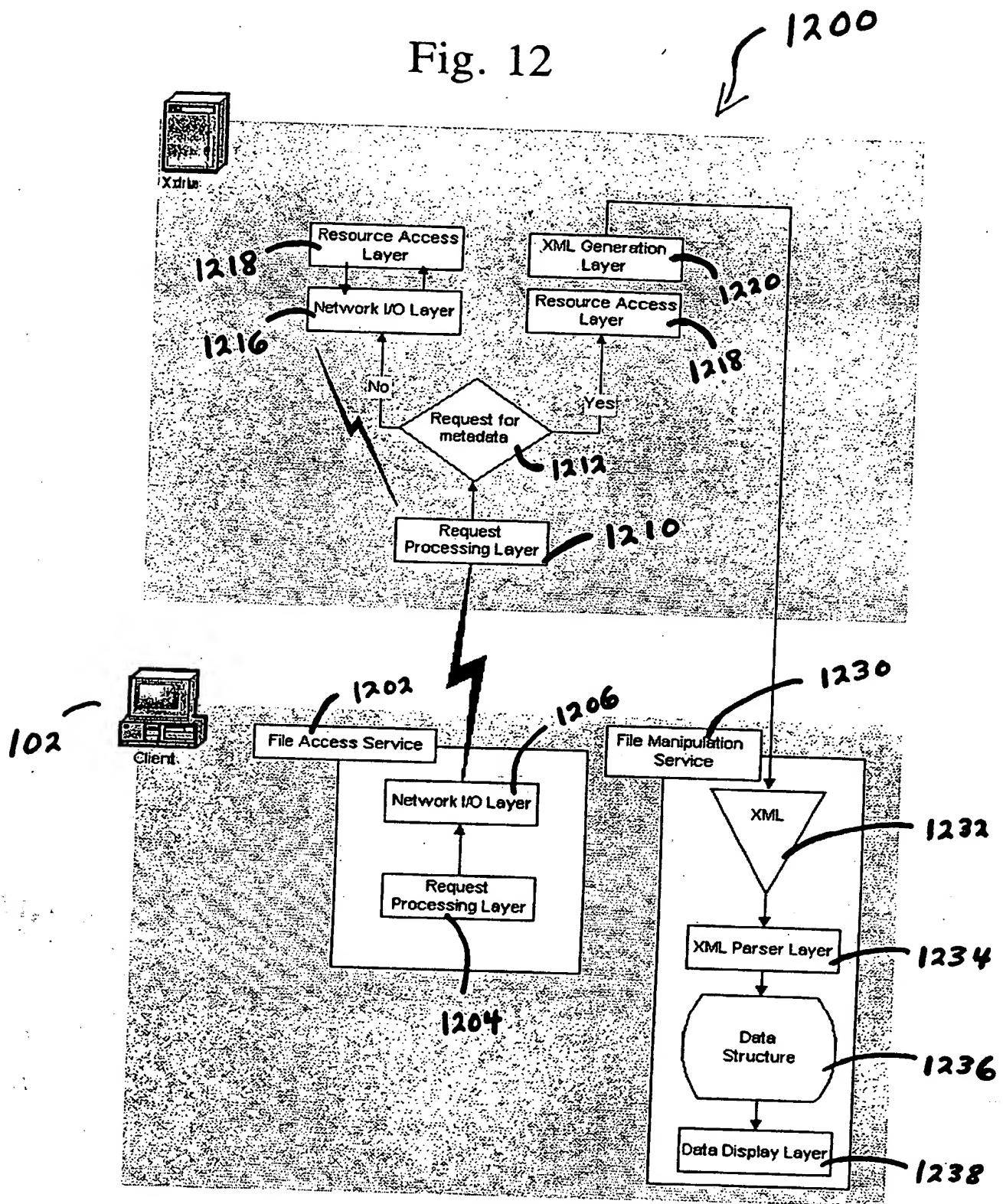


Fig. 13

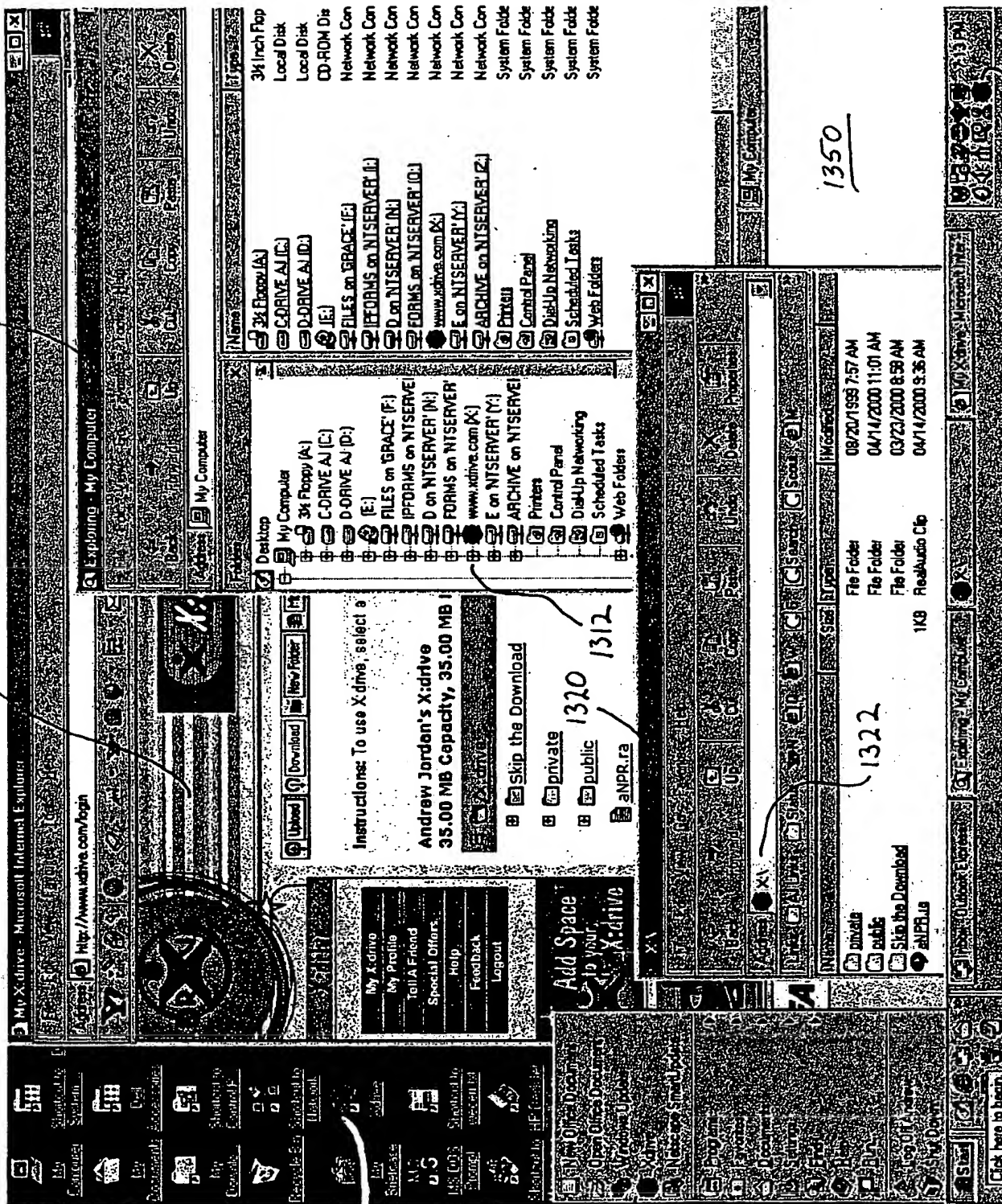
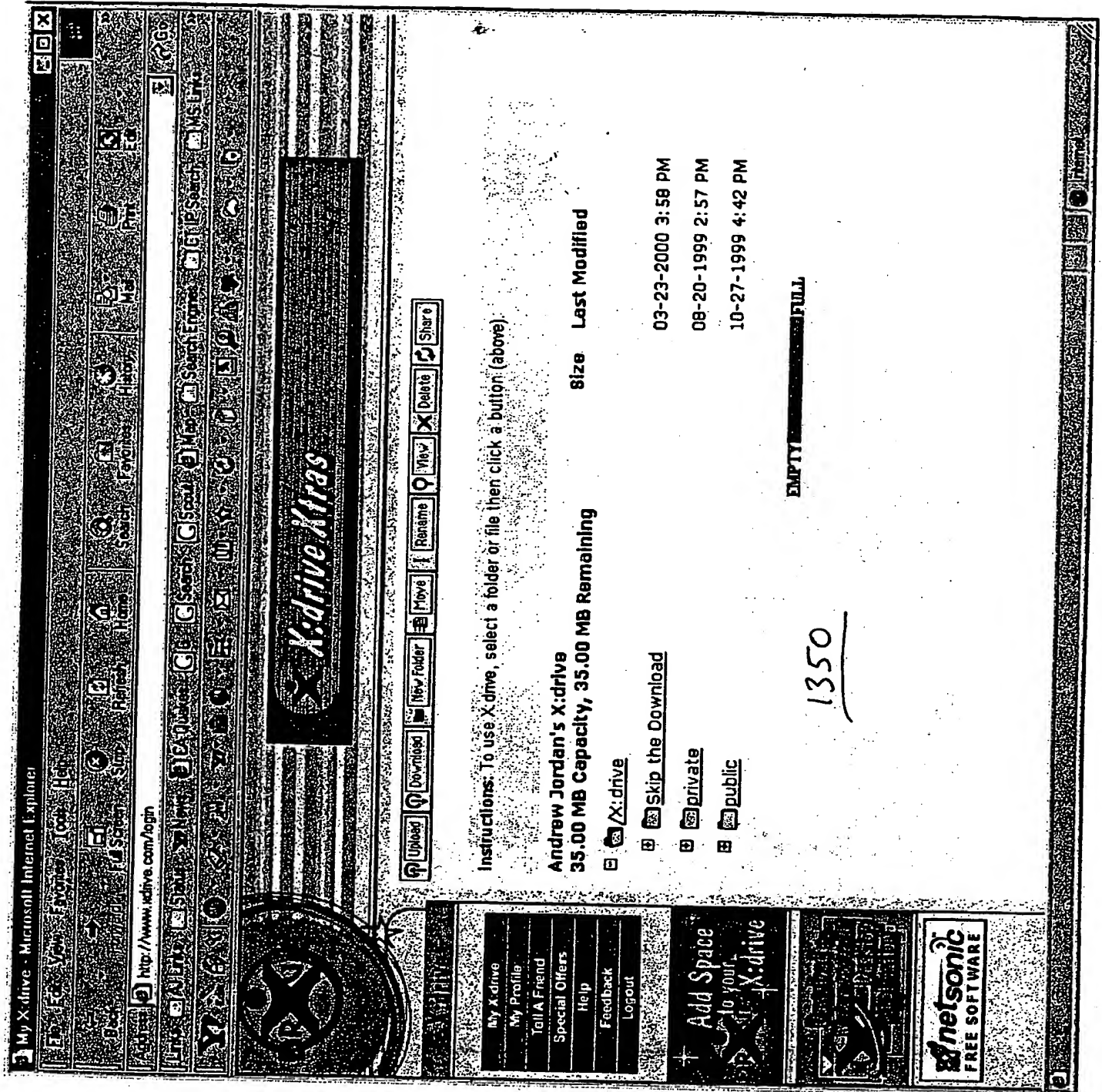


Fig. 14



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/30536**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(7) :G06F 15/00, 15/16, 17/30; B41B 15/00

US CL :345/326; 707/1,10; 709/104,105,212,213,217,226,229,245

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 345/326; 707/1,10; 709/104,105,212,213,217,226,229,245

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

East, West, IEEE

search terms : network, internet, storage, resource, parse, proxy

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,861,883 A (CUOMO et al.) 19 January 1999, col. 2 line 11 - col 6 line 37	1-50
Y	US 5,956,490 A (BUCHHOLTZ et al.) 21 September 1999, col. 2-6	1-50
Y,P	US 6,049,877 A (WHITE) 11 April 2000, col. 2 line 36 to col. 9 line 57	1-50
Y,E	US 6,154,738 A (CALL) 28 November 2000, col. 4 line 1 to col. 33 line 35	1-50
Y,E	US 6,151,601 A (PAPIERNIAK et al.) 21 November 2000, col. 8 line 35 to col. 25 line 67	1-50
Y,P	US 6,128,624 A (PAPIERNIAK et al.) 03 October 2000, col. 8 line 14 to col. 25 line 27.	1-50



Further documents are listed in the continuation of Box C.



See patent family annex.

*

Special categories of cited documents:

A

document defining the general state of the art which is not considered to be of particular relevance

E

earlier document published on or after the international filing date

L

document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

O

document referring to an oral disclosure, use, exhibition or other means

P

document published prior to the international filing date but later than the priority date claimed

T

later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

X

document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

Y

document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

&

document member of the same patent family

Date of the actual completion of the international search

10 JANUARY 2001

Date of mailing of the international search report

26 FEB 2001

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

FRANTZ B. JEAN

Telephone No. (703) 305-3900

```

## we need to do all of this to get the reseller code to show the
correct page
my $oDiskAccount = XDrive::DatabaseO::Table::DiskAccount->new(undef,
$oDBO);

$oDiskAccount->loadWhere("USERNAME", $username);

my $oUser = XDrive::DatabaseO::Table::UserData->new(undef,
$oDiskAccount->fetchDBO);

$oUser->loadWherePK($oDiskAccount->fetchColumn("USER_SEQ"));

my $oReseller = XDrive::DatabaseO::Table::Reseller->new(undef,
$oDiskAccount->fetchDBO);

$oReseller->loadWherePK($oUser->fetchColumn("RESELLER_SEQ"));

my $oTemplate = new XDrive::Template;

$oTemplate->partner($oReseller->fetchColumn("CODE"));

## originally this is where the signup_form.cgi goes
##$oTemplate->load('splash.thtml');
$oTemplate->load('tell_a_friend_frame.thtml');

##my $addrArray = $oCGI->param('friends_email_array');
##my $nameArray = $oCGI->param('friends_name_array');
##my $numFriends = $oCGI->param('numFriends');

## generate list for the javascript array
##my @addrList = split /,/, $addrArray;
##my @nameList = split /,/, $nameArray;

##$addrArray = "";
##$nameArray = "";

##my $count = @addrList - 1;

##for (my $i = 0;$i < $count;$i++) {
##    ##$addrArray .= "\"" . $addrList[$i] . "\", ";
##    ##$nameArray .= "\"" . $nameList[$i] . "\", ";
##}
## this will add the quote without the comma
##$addrArray .= "\"" . $addrList[$count] . "\"";
##$nameArray .= "\"" . $nameList[$count] . "\"";
## gets the array started
my $tempVar;

my $tempEmail = $oCGI->param('friends_email1');
my $numFriends = $oCGI->param('numFriends');

my $addrArray = "\"" . $tempEmail . "\"";
my $nameArray = "\"" . $oCGI->param('friends_name1') . "\"";

## generate list for the javascript array
for (my $i = 2;$i <= $numFriends;$i++)
{
    $tempVar = $oCGI->param('friends_email' . $i);

    if ($tempVar)
    {

```



```

$addrArray .= ", \" . $tempVar . "\"";
$nameArray .= ", \" . $oCGI->param('friends_name' . $i) . "\"";
}
}

$oTemplate->tags( ('numFriends' => $numFriends,
                  'friends_name_array' => $nameArray,
                  'friends_email_array' => $addrArray) );

print $oCGI->header();

print $oTemplate->get();

$oDiskAccount->finish();
$oUser->finish();
$oReseller->finish();
$oDiskAccount->disconnect();
}

#####
## Login in user who is coming from a Skip The Download
## Registration
#####

sub std_login () {
    my $username = shift;
    my $oCGI = shift;
    my $sSTDPartner = shift;
    my $sLanguage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sCatId = shift;
    my $sGid = shift;
    my $sSid = shift;
    my $oDBO = new XDrive::DatabaseO(undef);

    my $oError = new XDrive::Error;
    my $oToken = xd_login($oCGI, $username, $oError, $oDBO);
    xd_set_session_cookie($oCGI, $sSTDPartner, $sLanguage);

    my $oTemplate = new XDrive::Template
    (
        'partner_code' => $sSTDPartner,
        'language' => $sLanguage,
        'file' => 'skip_the_download_from_reg.shtml',
        'tags' =>
        (
            'FILE_URL' => $sFileURL,
            'FILE_NAME' => $sFileName,
            'ALTRUL' => $sAltURL,
            'LANG' => $sLanguage,
            'STDPARTNER' => $sSTDPartner,
            'CATID' => $sCatId,
            'GID' => $sGid,
            'SID' => $sSid,
        )
    );
};

```

```

    $oTemplate->clear();
    print "Content-type: text/html\n\n";
    print $oTemplate->get();
    $oDBO->disconnect();
}

sub contact_cybergold {
    my $oCGI = shift;
    my $msgid = shift;
    my $email = shift;

    my %args = (
        'mint_home'    => $ENV{'MINT_HOME'},
        'msg_mode'     => 'background_mode',

        'usr_email'    => $email,
        'msg_id'       => $msgid,

        'pay_type'     => 'reward',
        'pay_value'    => '1.00',
        'pay_readme'   => 'Thanks for registering with X:drive.',

        'co_name'      => 'X Drive',
        'co_key'       => 'registration',
        'co_account'   => '100500900000396',
        'mint_secret'  => '184FEB9DB81944502A1C91B2879484B6',

        'mint_url_pay' => 'http://www1.cybergold.com/payserver?pay_server',
        'msg_version'  => '2.2'
    );

    my($code, %res) = mint_invoke(\%args);

    ##this is temp code to print out stuff for cybergold
    ##my @keys = keys %res;
    ##my @values = values %res;
    ##while (@keys)
    ##{
    ##    die pop(@keys), '=', pop(@values), "\n";
    ##}

    return $code;
}

sub write_befree_log {
    my $oCGI = shift;

    my $source_id = $oCGI->cookie('sourceid');

    ##get the time
    ##needed to figure out name of file to write to
    my ($nSec, $nMin, $nHour, $nDay, $nMonth, $nYear, $sDay) =
        (localtime(time))[0,1,2,3,4,5,6];

    if ($nYear > 99) {
        $nYear = substr($nYear,1,2);
    }

    ## Numeric month is 0-11, so add one
    $nMonth++;

    ## Handle Y2K issue

```

```

        if ( $nYear >= 80 ) {
            $nYear += 1900;
        }
        else {
            $nYear += 2000;
        }

        my $dToday = sprintf("%s%02d%02d", $nYear, $nMonth, $nDay);
        my $dTodayFull = sprintf("%02d%02d%s%02d:%02d:%02d", $nMonth, $nDay, $nYear, $nHour, $nMin, $nSec);

        my $text =
"14524098\tS\t$dTodayFull\t$t$source_id\tl\tl\tl\t0.00\tUSD\tregistration\n";

        warn "#BF", $text, "\n";
        ##open(FILE, ">>xdrive_orders_$dToday.txt");
        ##print FILE $text;
        ##close(FILE);
    }

sub send_email_referee {
    my $user_seq = shift;
    my $oDBO = shift;
    my $oCookie = shift;
    my $additional_quota = shift;
    my $referred_from = shift;

    my $language = $oCookie->getElement('language');
    my $partner = $oCookie->getElement('partner');

    if ($language eq 'spanish') {
        my $text = 'un amigo que usted refirió';
        if ($referred_from eq '2') {
            $text = 'un usted compartió un fishero con';
        }
    }
    else {
        my $text = 'referred';
        if ($referred_from eq '2') {
            $text = 'shared a file with';
        }
    }

    my $text = 'referred';
    if ($referred_from eq '2') {
        $text = 'shared a file with';
    }

    ##comes in as k, change to megabytes
    my $mbs = $additional_quota/1024;

    my $oUserData = XDrive::DatabaseO::Table::UserData->new(undef,
$oDBO);
    $oUserData->loadWhere("SEQ", $user_seq);
    my $email_address = $oUserData->fetchColumn("EMAIL_ADDRESS");
    my $name_first = $oUserData->fetchColumn("NAME_FIRST");
    my $name_last = $oUserData->fetchColumn("NAME_LAST");

    my $oTemplate = new XDrive::Template( {'language' => $language,
'partner_code' => $partner} );

    $oTemplate->load('received_5MB_tellafriend.shtml');

```

```
$oTemplate->tags( {'mbs' => $mbs,
                  'text' => $text} );
$oTemplate->clear();

my $message = $oTemplate->get;

my %toXdrive =
(
    To      => "$name_first $name_last <$email_address>",
    Bcc     => '',
    From    => "support\@xdrive.com",
    Message => $message,
    Subject => "Congratulations!"
);

sendmail(%toXdrive);

$oUserData->finish();
}
```

###signup_form.cgi

```
#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999. Updated
## Fri Apr 5, 1996 to use new templates. Updated Wed Apr 21 1999 to use
## new library code.
```

```
use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp 'fatalsToBrowser';
use XDrive::CGI qw(:MAIN);
use XDrive::Client::Registration;
use XDrive::Template;
use XDrive::DatabaseO::Search;
use XDrive::Library;

use constant XD_REGISTRATION_DEFAULT_COUNTRY => 223;

exit &main;

sub main {
    my $oContent      = new XDrive::Template;
    my $oNavigation    = new XDrive::Template;
    my $oLayout        = new XDrive::Template;
    my $oCGI           = new CGI;

    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);

    my $oSearch;

    my $sReferee      = $oCGI->param('referee');
    my $sClaimTicket = $oCookie->getElement('ct');

    ## Defaults
    my $sUsername      = undef;
    my $sNameFirst     = undef;
    my $sNameLast      = undef;
    my $nYOB           = undef;
    my $nPromoSeq      = undef;
    my $nGender        = 3;
    my $sEmailAddress = undef;

    my ($country_seq, $occupation_seq, $postal_code, $ct_promo_seq);

    my %pullDownHash;
    if (XDDatabaseConnectionCheck() && XDNFSCheck())
    {
        $oSearch = XDrive::DatabaseO::Search->new(undef);
    }
    else
    {
        $sClaimTicket = undef;
        $oSearch = undef;
        %pullDownHash = generate_db_array();
    }
    if ($sClaimTicket) {
        my $rhData = getUserData($oSearch, $sClaimTicket);
```

```

    if ($rhData) {
        my $oNewCgi = CGI->new($rhData);

        $sUsername      = $oNewCgi->param('username');
        $sNameFirst      = $oNewCgi->param('name_first');
        $sNameLast       = $oNewCgi->param('name_last');
        $sEmailAddress   = $oNewCgi->param('email_address');
        $nYOB            = $oNewCgi->param('birth_year');
        $nGender         = $oNewCgi->param('gender');
        $soccupation_seq = $oNewCgi->param('occupation_seq');
        $scountry_seq    = $oNewCgi->param('country_seq');
        $postal_code     = $oNewCgi->param('postal_code');
    }

    if ($sReferee ne "") {
        # my $oCookie = XDrive::CGI::Cookie->new('x_session_info',
        $oCGI);

        my $sReferred_from = $oCGI->param('type');
        $oCookie->setElement({'partner_code'=>'xdrv'});
        $oCookie->setElement({'language'=>'english'});
        $oCookie->setElement({'referee' => $sReferee});
        $oCookie->setElement({'referred_from' => $sReferred_from});
        print "Set-Cookie: ".$oCookie->asString();
    }

    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');
    $oLayout->partner('xdrv');

    ## I'm assuming there will be one page and not a series of frames.
    ## this can be changed if need be
    # my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    # my $promo = $oCookie->getElement('promo');

    my $promo = $oCookie->getElement('promo');

    my $file_found;

    ##if we have a promo, try to get a special registration page
    if ($promo) {
        ##attempt to open a special registration page
        $file_found = $oLayout->load($promo . '_registration.shtml');
        if (!$file_found) {
            ##if we cannot, open the general promo reg page
            $file_found = $oLayout->load('promo_registration.shtml');
        }
    }

    ##is we don't have a promo then use the standard registration
    if ( (! $promo) || (! $file_found) ) {
        ## Load the required template HTML files.
        $oNavigation->load("front_nav.shtml");
        $oContent->load("front_signup.shtml");
        $oLayout->load("layout.shtml");

        $oContent->tags
        (
            'username'      => $sUsername,
            'name_first'    => $sNameFirst,
            'name_last'     => $sNameLast,
            'email_address' => $sEmailAddress,

```

```

        'country'          =>
xd_form_countries_db_check(XD_REGISTRATION_DEFAULT_COUNTRY,
$oSearCh,\%pullDownHash),
        'occupation'       => xd_form_occupation_db_check(undef,
$oSearCh,\%pullDownHash),
        'media_type'       => xd_form_media_type_db_check(undef,
$oSearCh,\%pullDownHash),
        'gender'           => xd_form_gender_db_check(undef,
$oSearCh,\%pullDownHash),
        'select_marketing' => 'CHECKED',
        'select_newsletter' => 'CHECKED',
        'referee'          => $sReferee,
    });

    ## Print out the HTML and exit
    $oLayout->tags
    (
        (
            'header_graphic' => 'header_registration.gif',
            'title' => 'Register Now!',
            'content' => $oContent->get,
            'navigation' => $oNavigation->get
        )
    )
    elsif ($sClaimTicket) {
        $oLayout->tags
        (
            (
                'country'          => xd_form_countries($country_seq,
$oSearCh),
                'occupation'       => xd_form_occupation($occupation_seq,
$oSearCh),
                'media_type'       => xd_form_media_type(undef, $oSearCh),
                'gender'           => xd_form_gender($nGender, $oSearCh),
                'select_marketing' => 'CHECKED',
                'select_newsletter' => 'CHECKED',
                'username'         => $sUsername,
                'name_first'       => $sNameFirst,
                'name_last'        => $sNameLast,
                'email_address'    => $sEmailAddress,
                'birth_year'       => $nYOB,
                'referee'          => $sReferee,
                'postal_code'      => $postal_code
            )
        )
    }
    else {
        $oLayout->tags
        (
            (
                'country'          =>
xd_form_countries_db_check(XD_REGISTRATION_DEFAULT_COUNTRY,
$oSearCh,\%pullDownHash),
                'occupation'       => xd_form_occupation_db_check(undef,
$oSearCh,\%pullDownHash),
                'media_type'       => xd_form_media_type_db_check(undef,
$oSearCh,\%pullDownHash),
                'gender'           => xd_form_gender_db_check(undef,
$oSearCh,\%pullDownHash),
                'select_marketing' => 'CHECKED',
                'select_newsletter' => 'CHECKED',
                'referee'          => $oCGI->param('referee'),
            )
        )
    }

    $oLayout->clear;

```

```

        print $oCGI->header, $oLayout->get;
        if (defined $oSearch)
        {
            $oSearch->disconnect();
        }

        return 0;
    }

## johngaa add to check of db is up or down
sub generate_db_array
{
    ## create a hash
    my %tempHash;
    my $i = 1;
    my $key;
    my @tempVal;
    open FH, "<down_data.dat";

    while(<FH>)
    {
        chomp $_;
        if ($_ =~ /^#(\w+)/g)
        {
            my @newArray;
            $i = 1;
            $key = $_;
            $tempHash{$key} = [ @newArray ];
        }
        else
        {
            @tempVal = split(/\~/, $_);

            $tempHash{$key}->[$i - 1][0] = $tempVal[0];
            $tempHash{$key}->[$i - 1][1] = $tempVal[1];
            $i++;
        }
    }

    close FH;
    return %tempHash;
}

sub xd_form_countries_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_countries(XD_REGISTRATION_DEFAULT_COUNTRY,
        $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $temp1 = $pullDownHash->{'country'};
        $returnVal = options_list(XD_REGISTRATION_DEFAULT_COUNTRY, @temp1);
    }
}

```



```
    return $returnVal;
}

sub xd_form_occupation_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_occupation(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'occupation'};
        $returnVal = options_list(undef, @$templ);
    }

    return $returnVal;
}

sub xd_form_media_type_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_media_type(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'media_type'};
        $returnVal = options_list(undef, @$templ);
    }

    return $returnVal;
}

sub xd_form_gender_db_check
{
    my $default = shift;
    my $oSearch = shift;
    my $pullDownHash = shift;
    my $returnVal;

    if (defined $oSearch)
    {
        $returnVal = xd_form_gender(undef, $oSearch),
    }
    else
    {
        ## insert alternate source of countries here
        my $templ = $pullDownHash->{'gender'};
        $returnVal = options_list(undef, @$templ);
    }
}
```

```
    return $returnVal;
}

## end of johngaa add
sub getPromoURI ($$) {
    my $oSearch = shift;
    my @promo_seq = (shift);

    my $oDBH = $oSearch->fetchDBO->fetchDBH();

    my $st = "SELECT uri FROM xdrive.promo WHERE seq = ?";

    my $data = $oDBH->selectcol_arrayref($st, undef, @promo_seq);

    return $data->[0];
}

sub getUserData {
    my $oSearch = shift;
    my $sTicket = shift;

    my $oDBH = $oSearch->fetchDBO->fetchDBH();
    my $sQuery = "SELECT DATA FROM BATCH_USER_DATA WHERE CODE = ?";
    my $oCursor = $oDBH->prepare($sQuery);
    $oCursor->bind_param(1, $sTicket);
    $oCursor->execute;

    my $rh;
    my $sData = $oCursor->fetchrow_array();
    # my ($sData) = $oCursor->fetchrow_array();
    # eval $sData;
    # return $rh;
    return $sData;
}
```

###signup_success.cgi

```
#!/usr/bin/perl
## This CGI allows us to pass the sst and sid on to the inner frame
##
## Modified by Justin White on 10/14/99 by manually printing the
## header to the browser and getting rid of the XDrive::CGI import.
## Created new cgi, database, and error objects to pass to xd_security_check.
## Also added the exit in the sub call.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI::Carp qw(fatalsToBrowser);
use CGI ();
use Token;
use XDrive::Client::Security;
use XDrive::Template;
use XDrive::DatabaseO;
use XDrive::Error;
use XDrive::Library;
use XDrive::CGI::Cookie;

&main();

exit;

sub main
{
    my $oCGI      = new CGI;
    my $oDBO      = new XDrive::DatabaseO;
    my $oErr      = new XDrive::Error;
    my $oCookie = new XDrive::CGI::Cookie('x_session_info', $oCGI);

    #####
    ## Attempt to authenticate the user
    #####

    my $oToken = xd_security_check($oDBO,$oCGI,$oErr);

    #####
    ## If the authentication fails or there is an error during the
    ## authentication phase then redirect to the error CGI
    #####

    if ($oErr->Occurud)
    {
        xd_fatal_error($oCGI,$oErr);
        exit;
    }

    #####
    ## Otherwise we have a valid session
    #####

    my $sUsername = $oToken->data('user');

    ### Edited by Justin so that the partner_code is looked for in
    ### the cookie instead of the token table.
    # my $sPartner = $oToken->data('partner_code');
    my $sPartner = $oCookie->getElement('partner');
```

```
    if (! defined $sPartner)
    {
        $sPartner = "xdrv";
        $oCookie->setElement({'partner'=>$sPartner});
        print "Set-Cookie: ", $oCookie->asString();
    }

    my $oTemplate = new XDrive::Template( {'partner_code' => $sPartner}
);

    $oTemplate->load('signup_success.thtml');
    $oTemplate->tags( {'username' => $sUsername} );

    print "content-type: text/html\n\n";

    print $oTemplate->get();

    $oDBO->disconnect();

    return 0;
}
```

###signup_toc.cgi

```

#!/usr/bin/perl
## Written by Martin Hald <mhald@uci.edu> on Sat, Jan 30, 1999. Updated
## Fri Apr 5, 1996 to use new templates.
##
## Modified by Justin White on 10/11/1999 so that it sets a cookie.
##
## Modified by Martin Hald on 11/15/1999 so that it now accepts
## - partner
## - language
## - agreeuri
## - disagreeuri

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI;
use CGI::Carp qw(fatalsToBrowser);
use XDrive::Template;
use XDrive::CGI::Cookie;

&main();

exit;

sub main {
    my $cookie;
    my $sPartnerCode;

    my $oCGI = new CGI;
    my $oCookie = XDrive::CGI::Cookie->new('x_session_info', $oCGI);
    my $sReferee = $oCGI->param('referee');
    my $sPartner = $oCGI->param('partner');
    my $sLanguage = $oCGI->param('language');
    my $sReferred_from = $oCGI->param('type');

    $oCookie->setElement({'partner_code'=>$sPartner});
    $oCookie->setElement({'language'=>$sLanguage});

    if ($sReferee ne "") {
        $oCookie->setElement({'referee' => $sReferee});
        $oCookie->setElement({'referred_from' => $sReferred_from});
        print "Set-Cookie: ".$oCookie->asString();
    }

    if (! defined $sPartner) {
        $sPartner = 'xdrv';
    }

    ## Load the terms and conditions
    my $hDefaults = {'partner_code'=>$sPartner, 'cookie'=>$oCookie};
    my $oContent = new XDrive::Template($hDefaults);
    my $oLayout = new XDrive::Template($hDefaults);

    $oContent->load('presignup.shtml');

    if ($sPartner eq 'xdrv') {
        my $oNavigation = new XDrive::Template($hDefaults);
        my $oHeader = new XDrive::Template($hDefaults);
        my $oFooter = new XDrive::Template($hDefaults);
    }
}

```

```
$oLayout->load('layout.shtml');
$oNavigation->load('front_nav.shtml');
$oHeader->load('presignup_header.shtml');
$oFooter->load('presignup_footer.shtml');
$oContent->tags({'header' => $oHeader->get,
               'footer'  => $oFooter->get, });
$oLayout->tags({'navigation' => $oNavigation->get,
               'header_graphic' => 'header_registration.gif',});
} else {
    $oLayout->load('tac_wrapper.shtml');
}

my $sAgreeURI    = $oCGI->param('agreeuri');
my $sDisagreeURI = $oCGI->param('disagreeuri');

$oLayout->tags({'title'      => 'Terms and Conditions',
               'content'    => $oContent->get,
               'agreeuri'   => $sAgreeURI,
               'disagreeuri' => $sDisagreeURI,});
$oLayout->clear;

print $oCGI->header();
print $oLayout->get;

return 0;
}
```

###skip_the_download.cgi

```

#!/usr/bin/perl

use strict;
use lib $ENV{PERL_XDRIVE_LIB};

use CGI qw(param redirect header cookie);
use CGI::Cookie;

use LWP::UserAgent;

use CGI::Carp qw(fatalsToBrowser);
use XDrive::Client::Security;
use XDrive::Client::Actions;
use XDrive::DatabaseO::Table::DiskAccount;
use XDrive::DatabaseO::Search;
use XDrive::DatabaseO::Transaction;
use XDrive::Template;
use XDrive::CGI qw(:MAIN);
use XDrive::CGI::Cookie;
use XDrive::DatabaseO;
use XDrive::Error;

use constant TRUE => (1==1);
use constant FALSE => ! TRUE;
use Token;

my $oDBO = new XDrive::DatabaseO;
main($oDBO);

$oDBO->disconnect;
exit;

#####
## NOTE: Remove the quota check from here. will be handled in java.
#####

sub main
{
    my $oDBO = shift;
    my $oCGI = CGI->new();
    my $oErr = new XDrive::Error;
    my $oCookie = XDrive::CGI::Cookie->new('xd_std_info', $oCGI);

    ## params for file url and file name
    my $sFileURL = $oCGI->param('FILEURL');
    my $sFileName = $oCGI->param('FILENAME');
    my $sAltURL = $oCGI->param('ALTURL');
    my $sSid = $oCGI->param('SID');
    my $sGid = $oCGI->param('GID');
    my $sCatId = $oCGI->param('CATID');
    my $sPartnerCode = $oCGI->param('STDPARTNER');
    my $sLanguageCode = $oCGI->param('LANG');
    my $sUsername = $oCGI->param('user');
    my $sPassword = $oCGI->param('pass');
    my $sError = $oCGI->param('error');
    my $sCookie = $oCGI->cookie('SST');

```

```

my $sessionCookie;
my $sPromo = '';
my $sPartnerParams = "";
my $sCNetString = "";

## IF THE SPECIAL C|NET VARIABLES ARE DECLARED
## THEN GENERATE THE C|NET STRING
## THIS URL IS CALLED FOR ANY FILE DOWNLOADED
## FROM C|NET SO THAT THEY CAN CREDIT THE FILE
## BEING DOWNLOADED
if (
    ($sSid != '') &&
    ($sGid != '') &&
    ($sCatId != '')
) {

    $sAltURL = "http://beta.cnet.com/downloads/0-" . $sCatId . "-107-"
. $sSid . ".html?tag=ex.dl.xdrive";

    ## IF YOU ARE ON THE TEST SERVERS,
    ## THEN USE C|NET'S TEST URL
    if (
        ($ENV{'HTTP_HOST'} eq 'martini.xdrive.com') ||
        ($ENV{'HTTP_HOST'} eq 'antifreeze.xdrive.com')
    ){

        $sCNetString = "http://abv-sjc2-
export2.cnet.com/downloads/0,10152,0-" .
            $sCatId .
            "-110-" .
            $sSid .
            ",00.html?gid=" .
            $sGid .
            "&tag=ex.dl.xdrivepop.dlcgi." .
            $sSid;

        ## ELSE, USE THEIR REAL URL
    } else {

        $sCNetString = "http://abv-sjc1-
export2.cnet.com/downloads/0,10152,0-" .
            $sCatId .
            "-110-" .
            $sSid .
            ",00.html?gid=" .
            $sGid .
            "&tag=ex.dl.xdrivepop.dlcgi." .
            $sSid;

    }

}

$sPartnerParams =
"STDPARTNER=$sPartnerCode&LANG=$sLanguageCode&ALTURL=$sAltURL";

$soCookie->setElement(
    {
        'FILEURL'      => $sFileURL,
        'FILENAME'     => $sFileName,
    }

```



```

        'ALTURL'      => $sAltURL,
        'STDPARTNER' => $sPartnerCode,
        'LANG'       => $sLanguageCode,
        'CATID'      => $sCatId,
        'SID'        => $sSid,
        'GID'        => $sGid,
    });

    print "Set-Cookie: ". $oCookie->asString();

    my $n = 0;
    my $rv;

    ## Create the database object
    my $oSearch = XDrive::Database0::Search->new($oDBO);

    ##The token for the user session
    my $oToken;

    ## If u/p
    if (defined $sUsername && defined $sPassword)
    {
        ## Auth or fail
        if (xd_auth_password($sUsername, $sPassword, $oDBO))
        {
            $oToken = xd_login($oCGI, $sUsername, $oErr);
            $sessionCookie = xd_set_session_cookie($oCGI,
            $sPartnerCode, $sLanguageCode, $sPromo);
        }
        else
        {
            ## Login failed
            my $r = getHTMLContent
            (
                'skip_the_download_login_failed.shtml',
                $sFileURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
            );

            print "Content-type: text/html\n\n";
            print $r;
            return 1;
        }
    }

    ## error or cookie not defined
    elsif ( (length($sError) > 0) || (length($sCookie) == 0) )
    {
        ## show the login page
        my $r = getHTMLContent('skip_the_download_login.shtml',
                                $sFileURL,
                                $sFileName,
                                $sAltURL,
                                $sPartnerCode,
                                $sLanguageCode
                            );

        print "Content-type: text/html\n\n";
        print $r;
    }

```

```

        return 1;
    }
    else
        ## cookie defined so authenticate it
        {
            $oToken = xd_security_check($oDBO,$oCGI,$oErr);
            $sessionCookie = xd_set_session_cookie($oCGI, $sPartnerCode,
            $sLanguageCode, $sPromo);

            if ($oErr->Occurud)
            {
                print $oCGI->redirect("/cgi-
bin/skip_the_download.cgi?&error=expired&$sPartnerParams");
                return 1;
            }
        }

    if (!$sFileURL) {
        my $thtml = ($sAltURL != '')?
'skip_the_download_no_alt_error.thtml'
                        : 'skip_the_download_error.thtml';

        my $sMessage = $oErr->ReturnMessageGivenCode(1220);

        &ThtmlErrorOut($thtml,
            $sMessage,
            $sFileURL,
            $sFileName,
            $sAltURL,
            $sPartnerCode,
            $sLanguageCode
        );
    }

    ## create the Actions object and download the file
    my $oAction = new XDrive::Client::Actions($oToken,$oCGI);

    ## set the filename and file url
    $oAction->STDFileName($sFileName);
    $oAction->STDURL($sFileURL);

    ## see if file exists. if yes, give em message
    my $bFileExists = $oAction->STDFileExists();

    if ($bFileExists)
    {
        $oDBO->disconnect();
        my $sMessage = $oErr->ReturnMessageGivenCode(1242);

        ErrorOut($sMessage,$sFileURL,$sFileName,$sAltURL,$sPartnerCode,$sLangua
geCode);
    }

    ## Check that the file is not already being downloaded
    if ($oSearch->XDSTDBeingDownloaded($oToken->user,$sFileURL))
    {
        $oDBO->disconnect();
        my $sMessage = $oErr->ReturnMessageGivenCode(1243);
    }

```

```

    ErrorOut ($sMessage,$sFileURL,$sFileName,$sAltURL,$sPartnerCode,$sLanguageCode);
}

## Spool the action to download the file
my $oTransaction = new XDrive::DatabaseO::Transaction($oDBO);
my $nSeq = $oTransaction->insertSkipTheDownload
(
    $oToken->user,
    $sFileName,
    $sFileURL,
    0,
    undef
);
$oTransaction->commit;

## Insert failed return an error
if ($nSeq < 0)
{
    $oDBO->disconnect();
    my $sMessage = $oErr->ReturnMessageGivenCode(1244);

    ErrorOut ($sMessage,$sFileURL,$sFileName,$sAltURL,$sPartnerCode,$sLanguageCode);
}

## IF THE INSERT DIDN'T FAIL,
## AND THE SPECIAL C\NET URL ISN'T NULL
## THEN CREDIT C\NET
elsif ($sCNetString ne '')
{
    my $oUA = new LWP::UserAgent;
    $oUA->agent("XDriveSTD/0.1 " . $oUA->agent);

    # Create a request
    my $oRequest = new HTTP::Request GET => $sCNetString;

    # Pass request to the user agent and get a response back
    my $oResult = $oUA->request($oRequest);
}

print redirect("/cgi-bin/skip_the_download_status.cgi?seq=$nSeq&$sPartnerParams");
}

sub ErrorOut ()
{
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

```

```

my $html = &getHTMLContent('skip_the_download_no_alt_error.html',
                           $sFileURL,
                           $sFileName,
                           $sAltURL,
                           $sPartnerCode,
                           $sLanguageCode,
                           $sMessage,
                           );

print "Content-type: text/html\n\n";
print $html;
exit(0);
)

```

```

sub ThtmlErrorOut ()
{
    my $thtml = shift;
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    my $html = &getHTMLContent($thtml,
                               $sFileURL,
                               $sFileName,
                               $sAltURL,
                               $sPartnerCode,
                               $sLanguageCode,
                               $sMessage,
                               );

    print "Content-type: text/html\n\n";
    print $html;
    exit(0);
}

```

```

sub getHTMLContent
{
    my $thtmlfile = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sMessage = shift;

    my $template = new XDrive::Template
    (
        'partner_code' => $sPartnerCode,
        'language' => $sLanguageCode,
        'file' => $thtmlfile,
        'tags' =>
        (
            'FILE_URL' => $sFileURL,
            'FILE_NAME' => $sFileName,
            'ALTURL' => $sAltURL,
            'LANG' => $sLanguageCode,

```

```
'STDPARTNER' => $sPartnerCode,
'message' => $sMessage,
}
));

$template->clear();

return $template->get;
}

## Create a string which makes the previously created
## cookie expire.

sub empty_cookie
{
    my $oSelf = shift;
    my $cookie = new CGI::Cookie
    (
        -name      => 'sst',
        -value     => '',
        -expires   => '-1M'
    );
    print header(-cookie=>[$cookie]);
}
```

###skip_the_download_status.cgi

```

#!/usr/bin/perl

use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header redirect);
use XDrive::CGI;
use XDrive::Client::Actions;
use XDrive::Client::Security;
use XDrive::DatabaseO;
use XDrive::DatabaseO::Table::SkipDownload;
use XDrive::Template;
use XDrive::Error;
use XDrive::Library;
use Token;

use strict;

use constant TEMP_DIR => XDSTDTempDirectory();

&main;
exit(0);

sub main
{
    ## get parameters
    my $nFileSize;
    my $sTempFile;
    my $sFileName;
    my $sError;
    my $nStatus;
    my $bDone;
    my $percent = 0;
    my $nDownloadedSize = 0;
    my $sURL;
    my $nNow;

    my $oCGI = new CGI();
    my $nSeq = $oCGI->param('seq');
    my $nStart = $oCGI->param('start');
    my $sPartnerCode = $oCGI->param('STDPARTNER');
    my $sLanguageCode = $oCGI->param('LANG');
    my $sAltURL = $oCGI->param('ALTURL');
    my $previous_percent = $oCGI->param('pp');

    ## SET THE CONNECTION_COUNT = 0 IF IT ISN'T PASSED IN
    my $connection_count = ($oCGI->param('cc')) ? $oCGI->param('cc') : 0;

    my $oErr = new XDrive::Error;

    ## get the token and the action object
    my $oDBO = new XDrive::DatabaseO;
    my $oToken = xd_security_check($oDBO, $oCGI, $oErr);
    my $oAction = new XDrive::Client::Actions($oToken, $oCGI);

    my $sPartnerParams =
"STDPARTNER=$sPartnerCode&LANG=$sLanguageCode&ALTURL=$sAltURL";

    if ($oErr->Occurud)

```

```

{
    print redirect("/cgi-bin/skip_the_download.cgi?$$PartnerParams");
    return;
}

## if the sequence number was passed then get information from the
database.
if (defined $nSeq)
{
    ## load the information from the database
    my $oSkip = XDrive::DatabaseO::Table::SkipDownload->new(undef, $oDBO);
    $oSkip->loadWhere('SEQ',$nSeq);
    $nFileSize = $oSkip->fetchColumn('FILE_SIZE_BYTES');
    $$TempFile = $oSkip->fetchColumn('FILENAME_FOR_TEMP_FILE');
    $$FileName = $oSkip->fetchColumn('FILE_NAME');
    $nStatus = $oSkip->fetchColumn('IS_ACTIVE');
    $$Error = $oSkip->fetchColumn('ERROR_CODE');
    $$URL = $oSkip->fetchColumn('FILE_URL');
    $bDone = $oSkip->fetchColumn('IS_DONE');
}

## XDRIVE.SKIP_THE_DOWNLOAD.IS_ACTIVE llegend
## 0 - still in queue
## 1 - being downloaded
## 2 - on hold

## IF CONNECTION_COUNT > 9, THEN GO TO THE FILE NOT FOUND (1220) ERROR
## DISPLAY, BUT KEEP TRYING TO DOWNLOAD THE FILE
if ($connection_count > 9) {
    $$Error=1220;
}

## IF AN ERROR OCCURRED THEN DISPLAY IT
## AND THEN EXIT(0);
if (defined $$Error)
{
    if ($$Error == 1240)
    {
        &DisplayQuotaError('',
            $$URL,
            $$FileName,
            $$AltURL,
            $$PartnerCode,
            $$LanguageCode
        );
    }
    else
    {
        my $oErr = new XDrive::Error;
        $oErr->AddErrorByErrorCode($$Error);
        &DisplayError($oErr->Message(),
            $$URL,
            $$FileName,
            $$AltURL,
            $$PartnerCode,
            $$LanguageCode
        );
    }
}

## IF THERE IS NO ERROR, THEN GATHER STATUS
## AND DISPLAY TO THE USER

```

```

else
(

## Get file size, later change to get from a tmp file
my $sPath = TEMP_DIR."/$$sTempFile";

## IF STATUS IS LISTED AS DONE IN THE DB,
## THEN SHOW THE DONE PAGE
if ($bDone == 1)
{
    &DisplayDone('',
                $sURL,
                $sFileName,
                $sAltURL,
                $sPartnerCode,
                $sLanguageCode
            );
}

## ELSE FILE IS NOT DONE,
## GATHER MORE DATA AND DISPLAY TO USER
else
{

    ## IF STATUS IS NOT ACTIVE, OR THE FILE DOESN'T EXIST
    ## THEN DISPLAY THE CONTACTING SERVER PAGE
    ## REMOVED: || ! -e $sPath
    ## FROM CHECK
    if ( ($nStatus == 0 || -e $sPath)
        && (!($previous_percent >= 0))
    )
    {

        &DisplayContactServer($nSeq,$sURL,$sFileName,$sAltURL,$sPartnerCode,$sL
anguageCode,$sPartnerParams,$connection_count);
    }

    ## ELSE, GATHER STATUS DATA
    ## AND DISPLAY TO USER
    else
    {

        ## Set the start time in seconds since the epoch if not passed
        ## as parameter
        if (! defined $nStart || $nStart !~ /\d+$/)
        {
            $nStart = time();
        }

        ## IF NO FILE SIZE HAS BEEN SET IN THE DB
        ## DISPLAY ZERO PERCENTAGES TO THE USER
        if (! defined $nFileSize || $nFileSize == 0)
        {
            $nFileSize = '0';
            $percent = '0';
            &DisplayStatus($nSeq,$percent,$sFileName,$nFileSize,'',
                $nStart,'','');
        }

        $sAltURL,$sPartnerCode,$sLanguageCode,$sPartnerParams);
    }
}

```



```

    }

    ## ELSE
    ## * THERE WAS NO ERROR
    ## * THE FILE WAS NOT DONE
    ## * THE FILE EXISTS IN THE TEMPORARY DIRECTORY
    ## * THE DB HAS AN EXPECTED FILE SIZE
    ## SO READ THE FILE, CALCULATE DATA, AND DISPLAY TO USER
    else
    {

        ## These checks are performed before inserting the skip
information    ## into the database, but we will do it again here to be
safe.

#       my $sError = $oErr->ReturnMessageGivenCode(141);
#       XDErrorToBrowser("", $sError, undef, $oToken);
        ##die "Cannot check $sPath" if $sPath =~ /\.\./;
        ##die "Cannot check $sPath" if $sPath =~ /\//;

        ## Get the size of the download object
        my @file_info = stat($sPath);

        ## Conver the downloaded file size into KB
        if ($file_info[7] > 0)
        {
            $nDownloadedSize = $file_info[7];

            if ($nFileSize > 0)
            {
                $percent = 100 * $nDownloadedSize/$nFileSize;
            }
            if ($percent < 0)
            {
                $percent = 0;
            }
            $percent = sprintf("%.2f", $percent);
        }

        ## IF THE FILE IS GONE NOW, OR SOMEOTHER CONDITION, THE USER
        ## WILL NEVER SEE THE %DONE DROP
        ## USE WHICH EVER IS LARGER, THE PRECENT THAT WE JUST
DISPLAYED      ## OF THE ONE THAT WE JUST READ FROM THE FILE SYSTEM
        $percent = ($previous_percent > $percent) ? $previous_percent
: $percent;

        ## We have already transfered some of the file, so we can now
        ## estimate the download time.
        $nNow = time();

        my $sInfo;
        my $nElapsedSec = $nNow - $nStart;
        my $nTransPerSec = 0;

        if ($nElapsedSec)
        {
            $nTransPerSec = $file_info[7]/$nElapsedSec;
        }

        if ($nTransPerSec > 0)

```

```

    {
        my $partial = $percent/100;
        my ($nSecsRemain, $nMin, $nSecs, $nTransPerSecMB);

        if ($partial == 0) {
            $sInfo = '';
        } else {
            $nSecsRemain = ($nElapsedSec/$partial)-$nElapsedSec;
            $nMin = int($nSecsRemain/60);
            $nSecs = $nSecsRemain % 60;
            $nTransPerSecMB = $nTransPerSec/1024;
        }

        $sInfo = sprintf(", %d:%02d remaining (%.2f
KB/sec)", $nMin, $nSecs
                                , $nTransPerSecMB);
    }

    my $nTrans;

    my $k = "KB";
    my $nDiv = 1024;
    my $nTempSize = $file_info[7] || 0;

    if ($nFileSize > 1024*1024)
    {
        $k = "MB";
        $nDiv = 1024*1024;
    }

    if ($nFileSize < 0)
    {
        $nFileSize = 0;
    }

    $nFileSize = sprintf("%.2f", $nFileSize/$nDiv);
    $nTrans = sprintf("%.2f", $nTempSize/$nDiv);

    &DisplayStatus($nSeq, $percent, $sFileName, $nFileSize, '',
        $nStart, $sInfo, $k,
        $sAltURL, $sPartnerCode, $sLanguageCode, $sPartnerParams);

    ## END OF READING DATA FROM SYSTEM AND
    ## DISPLAYING TO USER
    }

    ## END OF NO EXPECTED SIZE IN DB
    ## SHOW USER ZERO PERCENTAGES
    }

    ## END OF FILE MUST BE DONE
    ## SO SHOW A DONE
    }

    ## END OF NO ERROR
    }

    $oDBO->disconnect;
}

```

```

sub DisplayContactServer
{
    my
    ($nSeq, $sURL, $sFileName, $sAltURL, $sPartnerCode, $sLanguageCode, $sPartnerParams
    , $connection_count) = @_ ;

    my ($sHostname) = $sURL =~ /\:\/\/([^\\/]+)\//;
    $connection_count++;

    ## load the status page
    my $template = new XDrive::Template
    (
        'partner_code' => $sPartnerCode,
        'language' => $sLanguageCode,
        'file' => 'skip_the_download_contacting.thtml',
        'tags' =>
        (
            'hostname' => $sHostname,
            'continue_to' => "/cgi-
bin/skip_the_download_status.cgi?seq=$nSeq&cc=$connection_count&$sPartnerPara
ms",
            'fileName' => $sFileName,
            'altURL' => $sAltURL,
        )
    );
    print "Content-type: text/html\n\n";
    print $template->get;
}

```

```

sub DisplayStatus
{
    my $nSeq = shift;
    my $percent = shift;
    my $filename = shift;
    my $filesize = shift;
    my $transferred = shift;
    my $start = shift;
    my $info = shift;
    my $k = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;
    my $sPartnerParams = shift;

    my $percent_disp;

    if ($filesize <= 0)
    {
        $filesize = 'Unknown';
        $k = ' ';
        $percent_disp = 'Unknown';
        $percent = 0;
    }
    else
    {
        $percent_disp = "$percent%";
    }

    ## load the status page
    my $template = new XDrive::Template
    (

```

```

    'partner_code' => $sPartnerCode,
    'language' => $sLanguageCode,
    'file' => 'skip_the_download_status.shtml',
    'tags' =>
    {
        'PERCENT_DISP' => $percent_disp,
        'PERCENT' => $percent,
        'FILE_NAME' => $filename,
        'FILE_SIZE' => $filesize,
        'TRANSFERRED' => $transferred,
        'TRANSINFO' => $info,
        'K' => $k,
        'URL' => "/cgi-
bin/skip_the_download_status.cgi?seq=$nSeq&start=$start&pp=$percent&$sPartner
Params",
        'altURL' => $sAltURL
    }
});

$template->clear;
print "Content-type: text/html\n\n";
print $template->get;
}

```

```
sub DisplayDone
```

```

{
    my $sMessage = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    &ErrorOut('skip_the_download_complete.shtml',
        $sFileURL,
        $sFileName,
        $sAltURL,
        $sPartnerCode,
        $sLanguageCode,
        $sMessage
    );
}

```

```
sub DisplayError
```

```

{
    my $sError = shift;
    my $sFileURL = shift;
    my $sFileName = shift;
    my $sAltURL = shift;
    my $sPartnerCode = shift;
    my $sLanguageCode = shift;

    my $thtml = ($sAltURL != '')? 'skip_the_download_no_alt_error.shtml'
        : 'skip_the_download_error.shtml';

    &ErrorOut($thtml,
        $sFileURL,
        $sFileName,
        $sAltURL,
        $sPartnerCode,
    );
}

```

```

        $$LanguageCode,
        $$Error
    );

)

sub DisplayQuotaError
{
    my $$Error = shift;
    my $$FileURL = shift;
    my $$FileName = shift;
    my $$AltURL = shift;
    my $$PartnerCode = shift;
    my $$LanguageCode = shift;

    &ErrorOut('skip_the_download_quota_error.html',
        $$FileURL,
        $$FileName,
        $$AltURL,
        $$PartnerCode,
        $$LanguageCode,
        $$Error
    );
}

```

```

sub ErrorOut ()
{
    my $$THTMLFILE = shift;
    my $$FileURL = shift;
    my $$FileName = shift;
    my $$AltURL = shift;
    my $$PartnerCode = shift;
    my $$LanguageCode = shift;
    my $$Message = shift;

    my $template = new XDrive::Template
    (
        {
            'language' => $$LanguageCode,
            'partner_code' => $$PartnerCode,
            'file' => $$THTMLFILE,
            'tags' =>
            {
                'message' => $$Message,
                'altURL' => $$AltURL,
                'fileURL' => $$FileURL,
                'FILE_NAME' => $$FileName,
                'LANG' => $$LanguageCode,
                'ALTURL' => $$AltURL,
                'STDPARTNER' => $$PartnerCode,
            }
        }
    );

    my $html = $template->get;

    print "Content-type: text/html\n\n";
    print $html;
}

```



```

        $$LanguageCode,
        $$Error
    );

)

sub DisplayQuotaError
{
    my $$Error = shift;
    my $$FileURL = shift;
    my $$FileName = shift;
    my $$AltURL = shift;
    my $$PartnerCode = shift;
    my $$LanguageCode = shift;

    &ErrorOut('skip_the_download_quota_error.shtml',
        $$FileURL,
        $$FileName,
        $$AltURL,
        $$PartnerCode,
        $$LanguageCode,
        $$Error
    );
}

```

```

sub ErrorOut ()
{
    my $$THTMLFILE = shift;
    my $$FileURL = shift;
    my $$FileName = shift;
    my $$AltURL = shift;
    my $$PartnerCode = shift;
    my $$LanguageCode = shift;
    my $$Message = shift;

    my $template = new XDrive::Template
    (
        (
            'language' => $$LanguageCode,
            'partner_code' => $$PartnerCode,
            'file' => $$THTMLFILE,
            'tags' =>
            (
                'message' => $$Message,
                'altURL' => $$AltURL,
                'fileURL' => $$FileURL,
                'FILE_NAME' => $$FileName,
                'LANG' => $$LanguageCode,
                'ALTURL' => $$AltURL,
                'STDPARTNER' => $$PartnerCode,
            )
        )
    );

    my $html = $template->get;

    print "Content-type: text/html\n\n";
    print $html;
}

```

```

my $sUser_name = $oUserInfo->fetchColumn('NAME_FIRST') . " " .
$oUserInfo->fetchColumn('NAME_LAST');
my $sUser_email = $oUserInfo->fetchColumn('EMAIL_ADDRESS');
$oUserInfo->finish();
$oUserInfo->disconnect();

if ($sAddress)
{
    &send_mail($sName, $sAddress, $sUser_name, $sUser_email,
    $nUser_ID, $oCGI, $oToken, $oErr, $oCookie);
    &display_thank_you($oCGI, $oCookie);
}
else
{
    &display_form($oCGI, $oCookie);
}
}

sub send_mail {
    my ($sName, $sAddress, $sUser_name, $sUser_email, $nUser_ID, $oCGI,
    $oToken, $oErr, $oCookie) = @_ ;

    ## send out email for each friend only if form is filled out
    ## get number of friend fields

    my $numFriends = $oCGI->param("numFriends");
    for (my $i=1; $i<=$numFriends; $i++)
    {
        $sAddress = $oCGI->param('friends_email' . $i);
        $sName = $oCGI->param('friends_name' . $i);
        my $sMessage = &get_message($sUser_name, $nUser_ID, $sName,
        $sUser_email, $oCookie);

        ##only send the mail if the email address is filled out
        if ($sAddress)
        {
            my %toXdrive =
            (
                To      => "$sName <$sAddress>",
                Bcc      => "",
                From     => "$sUser_email",
                Message  => $sMessage,
                Subject  => "Check out X:drive!",
            );

            unless (sendmail %toXdrive)
            {
                warn "## Mail error ".$Mail::Sendmail::error;
                if ($Mail::Sendmail::error =~ /451/)
                {
                    my $sError = $oErr->ReturnMessageGivenCode(1310);
                    XDErrorToBrowser("", $sError, undef, $oToken);
                }
            }
            else
            {
                my $sError = $oErr->ReturnMessageGivenCode(1311);
                XDErrorToBrowser('tell_a_friend__error.shtml', $sError, undef, $oToken);
            }
            exit(1);
        }
    }
}

```


}

```

sub get_formfield {
    my ($sNum,$oCookie) = @_ ;

    my $oFormField = new XDrive::Template
    (
        'language'      => $oCookie->getElement('language'),
        'partner_code' => $oCookie->getElement('partner'),
    );
    $oFormField->load('tell_form_fields.thtml');

    $oFormField->tags
    (
        'number' => $sNum
    );

    return $oFormField->get;
}

sub get_message {
    my ($sUser_name, $nUser_ID, $sName, $sUserEmail,$oCookie) = @_ ;

    my $oMessage = new XDrive::Template
    (
        'language'      => $oCookie->getElement('language'),
        'partner_code' => $oCookie->getElement('partner'),
    );
    $oMessage->load('tell_a_friend_message.thtml');

    $oMessage->tags
    (
        'user_name' => $sUser_name,
        'nUser_ID' => $nUser_ID,
        'user_email' => $sUserEmail,
        'friend_name' => $sName
    );

    return $oMessage->get;
}

sub display_form {
    my $oCGI = shift;
    my $oCookie = shift;
    my $oForm = new XDrive::Template
    (
        'language'      => $oCookie->getElement('language'),
        'partner_code' => $oCookie->getElement('partner'),
    );
    $oForm->load('tell_a_friend.thtml');
    my $numFriends = $oCGI->param("numFriends");

    ##construct the html for multiple input fields
    my $inputFields='';

    for (my $i=1; $i<=$numFriends ; $i++)
    {
        $inputFields = $inputFields . &get_formfield($i,$oCookie);
    }
}

```

```
}

$soForm->tags
  (
    'friendsToTell' => $inputFields,
    'numFriends' => $numFriends,
  );
print $oCGI->header, $soForm->get;
exit(0);
}

sub display_thank_you {
  my $oCGI = shift;
  my $oCookie = shift;
  my $soForm = new XDrive::Template
    (
      'language' => $oCookie->getElement('language'),
      'partner_code' => $oCookie->getElement('partner'),
    );
  $soForm->load('tell_a_friend__t_y.thtml');
  print $oCGI->header, $soForm->get;
  exit(0);
}
```

###web_unauthorized.cgi

```

#!/usr/bin/perl
# Written by Martin Hald <mhald@uci.edu> on Sat Feb 13, 1999
#
# Program for showing unauthorized information and allowing the users to
# re-login and possibly showing them a "forgot your password?" link.

use strict;
use lib ($ENV{PERL_XDRIVE_LIB});

use CGI qw(header param);
use CGI::Carp qw(fatalsToBrowser);
# use XDrive::CGI qw(:MAIN);
use XDrive::Client::Registration;
use XDrive::Template;
use XDrive::Error;

exit &main;

sub main
{
    my $oCGI = CGI->new();

    my $oLayout = new XDrive::Template;
    my $oContent = new XDrive::Template;
    my $oNavigation = new XDrive::Template;

    $oLayout->partner('xdrv');
    $oContent->partner('xdrv');
    $oNavigation->partner('xdrv');

    $oLayout->load('layout.thtml');
    $oNavigation->load('front_nav.thtml');

    ## Get the error key
    my $sError = $oCGI->param('error');

    ##now get the error message associated with that error
    my $oErr = new XDrive::Error;
    my $message = $oErr->ReturnMessageGivenCode($sError);

    ## Load the required template HTML files.
    my $oForm = new XDrive::Template;
    $oForm->partner('xdrv');
    $oForm->load("front_nav.thtml");
    $oContent->load("unauthorized.thtml");

    ## Update the layout
    $oLayout->tags
    (
        (
            'header_graphic' => 'header_denied.gif'
        )
    );

    ## Update the content
    $oContent->tags
    (
        (
            'error_message' => $message
        )
    );
    $oContent->clear();
}

```

```
## Print out the HTML and exit
$oLayout->tags
({
  'content' => $oContent->get,
  'navigation' => $oNavigation->get,
  'title' => 'Authorization Denied'
});
print header(), $oLayout->get;

return 0;
```

)

Windows Client Code

// Module: dlgShareAFile.h	1
// Module: dlgShareAFile.h	3
// Module: xdBase64.cpp	5
// Module: xdBase64.h	9
// Module: xdGlobals.h	10
// Module: xdParseDate.h	13
// Module: xdRegistry.h	14
// Module: xdTokens.h	16
// Module: xdTools.h	17
// Module: xdEngine.h	20
// Module: tdimsgtbl.h	22
// Module: tdisock.h	24
// Module: xdFileIO.cpp	41
// Module: xdDebugger.cpp	45

//

// Module: dlgShareAFile.h

// Subsystem: KnoWare Internet Engine (kwEngine.dll)
 // Contents: Declaration module for the dlgShareAFile class.

//

// -----
 // Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.

//

// -----
 //

#include "stdafx.h"
#include <xdGlobals.h>

#ifndef _VXD_SOURCE_
#include "resource.h"
#endif
#include "dlgShareAFile.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// -----
// Implementation
// -----

BEGIN_MESSAGE_MAP(dlgShareAFile, CDialog)
 //{{AFX_MSG_MAP(dlgShareAFile)
 //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// -----
// Method: dlgShareAFile()
// Purpose: Standard constructor
//

dlgShareAFile::dlgShareAFile(CWnd* pParent /*=NULL*/) : CDialog(dlgShareAFile::IDD, pParent)

{
 //{{AFX_DATA_INIT(dlgShareAFile)
 m_sFileName = szEMPTY;
 m_sFileDescription = szEMPTY;
 m_sEmailMessage = szEMPTY;
 m_sEmailSubject = szEMPTY;
 m_sEmail0 = szEMPTY;
 m_sEmail1 = szEMPTY;
 m_sEmail2 = szEMPTY;
 m_sEmail3 = szEMPTY;
 m_sEmail4 = szEMPTY;
 //}}AFX_DATA_INIT
} // End of dlgShareAFile()

// -----
// Method: DoDataExchange()
// Purpose: Standard data exchange handler
//

void dlgShareAFile::DoDataExchange(CDataExchange* pDX)
{

```

    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(dlgShareAFile)
    DDX_Text(pDX, IDC_SHARE_FILENAME, m_sFileName);
    DDX_Text(pDX, IDC_SHARE_FILEDESC, m_sFileDescription);
    DDX_Text(pDX, IDC_SHARE_EMAILMSG, m_sEmailMessage);
    DDX_Text(pDX, IDC_SHARE_EMAILSUB, m_sEmailSubject);
    DDX_Text(pDX, IDC_SHARE_EMAIL1, m_sEmail0);
    DDX_Text(pDX, IDC_SHARE_EMAIL2, m_sEmail1);
    DDX_Text(pDX, IDC_SHARE_EMAIL3, m_sEmail2);
    DDX_Text(pDX, IDC_SHARE_EMAIL4, m_sEmail3);
    DDX_Text(pDX, IDC_SHARE_EMAIL5, m_sEmail4);
   //}}AFX_DATA_MAP
} // End of DoDataExchange()

```

```

// -----
// Method: OnInitDialog()
// Purpose: Called to initialize the contents of the dialog
//
BOOL dlgShareAFile::OnInitDialog()
{
    CDialog::OnInitDialog();

    UpdateData(FALSE);
    return TRUE; //return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
} // End of OnInitDialog()

```

```

// -----
// Method: OnOK()
// Purpose: Called to close out the dialog.
//
void dlgShareAFile::OnOK()
{
    UpdateData(TRUE);
    CDialog::OnOK();
} // End of OnOK()

```

//

// Module: dlgShareAFile.h

// Subsystem: KnoWare Internet Engine (kwEngine.dll)
 // Contents: Declaration module for the dlgShareAFile class.

//

// -----
 // Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.

//

// -----
 //

#if !defined(INC_DLGSHAREAFIle_H_)
 #define INC_DLGSHAREAFIle_H_

#if _MSC_VER > 1000
 #pragma once
 #endif // _MSC_VER > 1000

#ifndef _VXD_SOURCE_
 #include "resource.h"
 #endif

#ifndef _VXD_SOURCE_

// -----

// dlgShareAFile dialog class

//

class dlgShareAFile : public CDialog

{

public:

dlgShareAFile(CWnd* pParent = NULL); // standard constructor

//{AFX_DATA(dlgShareAFile)

enum { IDD = IDD_SHARE };

CString m_sFileName;

CString m_sFileDescription;

CString m_sEmailMessage;

CString m_sEmailSubject;

CString m_sEmail0;

CString m_sEmail1;

CString m_sEmail2;

CString m_sEmail3;

CString m_sEmail4;

//}AFX_DATA

//{AFX_VIRTUAL(dlgShareAFile)

protected:

virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support

//}AFX_VIRTUAL

protected:

//{AFX_MSG(dlgShareAFile)

virtual BOOL OnInitDialog();

virtual void OnOK();

//}AFX_MSG

DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}

// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif

#endif // !defined(_INC_DLGSHAREAFILE_H)

//

// **Module: xdBase64.cpp**// Subsystem: X:drive Client Engine (xdEngine.dll)
// Contents: Implementation module for the xdBase64 class

//

// -----
// Copyright (c) 1999 by X:drive(tm), Inc.
// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
// All rights reserved.

//

//

//

#include "stdafx.h"
#include "xdBase64.h"#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
#ifdef _VXD_SOURCE_
#include <xdEngine.h>
#define TRACE_DEBUG_DPRINTF
#endif

// Static Member Initializers

//

// The 7-bit alphabet used to encode binary information
CString xdBase64::m_sBase64Alphabet =
_T("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/");

int xdBase64::m_nMask[] = { 0, 1, 3, 7, 15, 31, 63, 127, 255 };// -----
// Method: xdBase64()
// Purpose: Standard Constructor
//
xdBase64::xdBase64 (void)
{
} // End of xdBase64()// -----
// Method: ~xdBase64()
// Purpose: Standard destructor
//
xdBase64::~~xdBase64()
{
} // End of ~xdBase64()// -----
// Method: Encode()
// Purpose: Encodes a string
//
CString xdBase64::Encode(LPCTSTR szEncoding, int nSize)
{
 CString sOutput = _T("");
 int nNumBits = 6;
 UINT nDigit;
 int lp = 0;

```

    ASSERT( szEncoding != NULL );
    if( szEncoding == NULL )
        return sOutput;
    m_szInput = szEncoding;
    m_nInputSize = nSize;

    m_nBitsRemaining = 0;
    nDigit = read_bits( nNumBits, &nNumBits, lp );
    while( nNumBits > 0 )
    {
        sOutput += m_sBase64Alphabet[ (int)nDigit ];
        nDigit = read_bits( nNumBits, &nNumBits, lp );
    }
    // Pad with '=' as per RFC 1521
    while( sOutput.GetLength() % 4 != 0 )
    {
        sOutput += '=';
    }
    return sOutput;
} // End of Encode()

// -----
// Method: Decode()
// Purpose: Decodes data
// Notes: The size of the output buffer must not be less than 3/4 the
//        size of the input buffer. For simplicity, make them the same
//        size.
//
int xdBase64::Decode(LPCTSTR szDecoding, LPTSTR szOutput)
{
    CString sInput;
    int c, lp=0;
    int nDigit;
    CString strDecode;
    int* pDecode = (int*)strDecode.GetBuffer(256*sizeof(int));

    ASSERT( szDecoding != NULL );
    ASSERT( szOutput != NULL );
    if( szOutput == NULL )
        return 0;
    if( szDecoding == NULL )
        return 0;
    sInput = szDecoding;
    if( sInput.GetLength() == 0 )
        return 0;

    // Build Decode Table
    //
    for( int i = 0; i < 256; i++ )
        pDecode[i] = -2; // Illegal digit
    for( i=0; i < 64; i++ )
    {
        pDecode[ m_sBase64Alphabet[ i ] ] = i;
        pDecode[ m_sBase64Alphabet[ i ] | 0x80 ] = i; // Ignore 8th bit
        pDecode[ '=' ] = -1;
        pDecode[ '=' | 0x80 ] = -1; // Ignore MIME padding char
    }

    // Clear the output buffer
    memset( szOutput, 0, sInput.GetLength() + 1 );

    // Decode the Input

```

```

//
for( lp = 0, i = 0; lp < sInput.GetLength(); lp++ )
{
    c = sInput[ lp ];
    nDigit = pDecode[ c & 0x7F ];
    if( nDigit < -1 )
    {
        return 0;
    }
    else if( nDigit >= 0 )
        // i (index into output) is incremented by write_bits()
        write_bits( nDigit & 0x3F, 6, szOutput, i );
}

return i;
} // End of Decode()

// -----
// Method: read_bits()
// Purpose: dunno
//
UINT xdBase64::read_bits(int nNumBits, int * pBitsRead, int& lp)
{
    ULONG lScratch;
    while( ( m_nBitsRemaining < nNumBits ) &&
           ( lp < m_nInputSize ) )
    {
        int c = m_szInput[ lp++ ];
        m_lBitStorage <<= 8;
        m_lBitStorage |= (c & 0xff);
        m_nBitsRemaining += 8;
    }
    if( m_nBitsRemaining < nNumBits )
    {
        lScratch = m_lBitStorage << ( nNumBits - m_nBitsRemaining );
        *pBitsRead = m_nBitsRemaining;
        m_nBitsRemaining = 0;
    }
    else
    {
        lScratch = m_lBitStorage >> ( m_nBitsRemaining - nNumBits );
        *pBitsRead = nNumBits;
        m_nBitsRemaining -= nNumBits;
    }
    return (UINT)lScratch & m_nMask[nNumBits];
} // End of read_bits()

// -----
// Method: write_bits()
// Purpose: dunno
//
void xdBase64::write_bits ( UINT nBits, int nNumBits, LPTSTR szOutput, int& i )
{
    UINT nScratch;

    m_lBitStorage = (m_lBitStorage << nNumBits) | nBits;
    m_nBitsRemaining += nNumBits;
    while( m_nBitsRemaining > 7 )
    {
        nScratch = m_lBitStorage >> ( m_nBitsRemaining - 8 );
        szOutput[ i++ ] = (TCHAR)(nScratch & 0xFF);
        m_nBitsRemaining -= 8;
    }
}

```

WO 01/33381
} // End of write_bits()

PCT/US00/30536

//

// **Module: xdBase64.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: Declaration module for the xdBase64 class.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnowWare(r), Inc.

// All rights reserved.

//

// -----

//

#if !defined(_INC_XDBASE64_H_)

#define _INC_XDBASE64_H_

#ifdef _VXD_SOURCE_

#include <xdCString.h>

#endif

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

// -----

// xdBase64 encoder class

//

class xdBase64

{

public:

xdBase64 (void);

virtual ~xdBase64 (void);

virtual int

Decode (LPCTSTR szDecoding, LPTSTR szOutput);

virtual CString Encode (LPCTSTR szEncoding, int nSize);

protected:

void

write_bits (UINT nBits, int nNumBts, LPTSTR szOutput, int& lp);

UINT

read_bits (int nNumBits, int* pBitsRead, int& lp);

protected:

int

m_nInputSize;

int

m_nBitsRemaining;

ULONG

m_lBitStorage;

LPCTSTR

m_szInput;

static int m_nMask[];

static CString m_sBase64Alphabet;

};

#endif // !defined(_INC_XDBASE64_H_)

// Module: xdGlobals.h

// Subsystem: X:drive

// Contents: Global definitions used throughout the system

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

// -----

// -----

//

#ifndef _INC_XDGLOBSALS_H_

#define _INC_XDGLOBSALS_H_

#ifdef _VXD_SOURCE_

//

// This HodgePodge helps us to be able to compile all of our code

// under Ring-3 and Ring-0 without too much modification.

//

#ifndef USE_NDIS

#define USE_NDIS

#endif

#include <vtoolscp.h>

// VToolsD main header file

#ifndef LPCTSTR

typedef char TCHAR;

typedef unsigned char TUCCHAR;

typedef const TCHAR* LPCTSTR;

typedef TCHAR* LPTSTR;

typedef unsigned char BYTE;

typedef BYTE* LPBYTE;

typedef DSKTLSYSTEMTIME SYSTEMTIME;

typedef HANDLE HINSTANCE;

#define _T(x) (x)

#endif

#ifndef BASED_CODE

#define BASED_CODE

#endif

#ifndef INVALID_HANDLE_VALUE

#define INVALID_HANDLE_VALUE (HANDLE)-1

#endif

#define _tcsstr strstr

// Standard unicode mappings

#define _tcslen strlen

#define _tcsncpy strcpy

#define _tcsrchr strrchr

#define _tcscat strcat

#define _ttoi atoi

#define _ttoi atol

#define _tcsrev strrev

#define _tcschr strchr

#define _tcsncpy strncpy

#define _tcsprbrk strpbrk

#define _tprintf sprintf

#define _tcslwr strlwr

```

#define _tcsupr      strupr
#define _tcsicmp     stricmp
#define _tcscmp      strcmp
#define _tcscoll     strcmp
#define _istdigit    isdigit
// #define ASSERT Assert
typedef HANDLE      HWND;
#endif

// -----
// Setup a whole bunch of constants that we can use throughout the systems
//
#define chNL          _T("\n")
#define chCOMMA       _T(',')
#define chDOSSLASH    _T("\")
#define chUNIXSLASH   _T("/")
#define chQUOTE       _T("\"")
#define chDQUOTE      _T("\'")
#define chPERIOD      _T('.')
#define chBAR         _T('|')
#define chTAB         _T("\t")
#define chCR          _T("\r")
#define chSPACE       _T(' ')
#define chCOLON       _T(':')
#define chSEMICOLON   _T(';')
#define chDASH        _T('-')
#define chPLUS        _T('+')
#define chPERCENT     _T('%')
#define chOPENBRACKET _T '['
#define chCLOSEBRACKET _T ']'
#define chNUL         _T('\0')
#define chZERO        _T('0')
#define chONE         _T('1')
#define chTWO         _T('2')
#define chTHREE       _T('3')
#define chFOUR        _T('4')
#define chFIVE        _T('5')
#define chSIX         _T('6')
#define chSEVEN       _T('7')
#define chEIGHT       _T('8')
#define chNINE        _T('9')
#define chOPENPAREN   _T('(')
#define chCLOSEPAREN  _T(')')
#define chAT          _T('@')

#define szNL          _T("\n")
#define szCOMMA       _T(",")
#define szDOSSLASH    _T("\")
#define szUNIXSLASH   _T("/")
#define szQUOTE       _T("\"")
#define szDQUOTE      _T("\'")
#define szPERIOD      _T(".")
#define szBAR         _T('|')
#define szTAB         _T("\t")
#define szCR          _T("\r")
#define szSPACE       _T(" ")
#define szCOLON       _T(":")
#define szSEMICOLON   _T(";")
#define szDASH        _T("-")
#define szPLUS        _T("+")
#define szOPENBRACKET _T "["
#define szCLOSEBRACKET _T "]"

```



```

#define szAT _T("@")
#define szEMPTY _T("")
#define szCURRENTDIR _T(".")
#define szPARENTDIR _T("..")
#define szFTP_DOT _T("ftp.")
#define szFTP_SLASH _T("ftp://")
#define szOPENPAREN _T("(")
#define szCLOSEPAREN _T(")")

#define XD_CACHE_BASEDIR _T("xdcache")
#define XD_LOGFILE_NP _T("xdrive.log")
#define XD_LOGFILE_VXD _T("xdrivevxd.log")

// -----
// We need to define the scope of values which will be used in the system.
// They are defined here since we need to read/write these to the registry.
//

//
// General defines
//
#define XD_LEN_32 32
#define XD_LEN_64 64
#define XD_LEN_128 128
#define XD_LEN_256 256
#define XD_LEN_512 512
#define XD_LEN_1024 1024
#define XD_LEN_2048 2048

//
// these program IDs are also the 1st two digits of the registration number
//
#define XD_PROGID_XDRIVE 0x53 // {DB2112AD-0000-0000-0053-000004281965}

//
// IN will generate a directory listing and the local file that contains
// that information will have an extension of '.fnd'. For example, if
// IN/FND does a directory listing of ftp.microsoft.com/softlib/mslfiles,
// it will place the raw directory listing in the in the local IN cache
// directory (which is currently defined as hanging off of the same
// directory where IN is located) as
//
// c:\xdCache\ftp.microsoft.com\root.softlib.mslfiles.ls
//
// and the parsed FND formatted data will be placed into
//
// c:\xdCache\ftp.microsoft.com\root.softlib.mslfiles.fnd
//
// the .fnd file is parsed out to produce the information returned as a
// result of the FINDFIRST()/FINDNEXT() calls to the NP.
//
#define XD_FILEEXT_LS _T(".ls")
#define XD_FILEEXT_XDR _T(".fnd")
//
// Here is our Network Provider Name
//
#define XD_PROVIDER_NAME _T("Xdrive")
#define XD_PROVIDER_NETID 0x00120000

#endif // _INC_XDGLOBALS_H_

```

//

// **Module: xdParseDate.h**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Declaration module for the CParseDate utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnowWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDPARSEDATE_H_

#define _INC_XDPARSEDATE_H_

#include <xdTokens.h>

class XDTOOLS_PUBLIC CParseDate

{

public:

CParseDate (void);

~CParseDate (void);

BOOL Parse (LPCTSTR s);

int	m_iYear;
int	m_iMonth;
int	m_iDay;
int	m_iHour;
int	m_iMinute;
int	m_iSecond;

TCHAR m_szDate[64];

TCHAR m_szTime[32];

TCHAR m_szOrig[64];

private:

BOOL isNUM (LPCTSTR s);

BOOL isDOW (LPCTSTR s);

xdTokens m_tokens;

};

#endif

//

// Module: xdRegistry.h

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Declaration module for the xdRegistry utility class

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#ifndef _INC_XDREGISTRY_H_

#define _INC_XDREGISTRY_H_

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

#include <xdGlobals.h>

// X:drive system wide globals

#include <xdTools.h>

// X:drive Tools Related

// -----

// xdRegistry

// the registry class encapsulates the registry functions. You must open

// at least a hive in the constructor. then you can optionally open

// a subkey & read/write information to the registry. All methods will return

// true upon successful completion. false will be returned if an error

// has occurred.

//

class XDTOOLS_PUBLIC xdRegistry

{

public:

xdRegistry();

~xdRegistry();

//

// public interface

//

public:

BOOL RegOpenRead (HKEY hHive, LPCTSTR szSubKey);

BOOL RegOpenWrite (HKEY hHive, LPCTSTR szSubKey);

BOOL RegClose (void);

BOOL RegDeleteKey (HKEY hHive, LPCTSTR szSubKey);

BOOL RegDeleteValue (LPCTSTR szVal);

BOOL RegEnumKey (int i, LPCTSTR szKeyName, UINT uiLenWithNull);

BOOL RegEnumVal (int i, LPCTSTR szValName, UINT uiLenWithNull, LPCTSTR
szValData, UINT uiDataLenWithNull);

BOOL RegEnumStr (int i, LPCTSTR szVal, UINT uiLenWithNull);

BOOL RegGetStr (LPCTSTR sName, LPCTSTR szVal, UINT uiLenWithNull);

BOOL RegPutStr (LPCTSTR sName, LPCTSTR szVal);

BOOL RegPutBin (LPCTSTR sName, BYTE* pBuffer, UINT uiLen);

BOOL RegGetNum (LPCTSTR sName, BOOL& bVal);

BOOL RegGetNum (LPCTSTR sName, WORD& wVal);

BOOL RegGetNum (LPCTSTR sName, DWORD& dwVal);

BOOL RegGetNum (LPCTSTR sName, UINT& uiVal);

BOOL RegPutNum (LPCTSTR sName, DWORD dwVal);

```
        LONG          RegGetLastError ( void );

private:
    HKEY    m_hKey;        // the current open hive
    LONG    m_lRetCode;    // the last return code
}; // End of xdRegistry

#endif // _INC_XDREGISTRY_H_
```

//

// Module: xdTokens.h

// Subsystem: X:drive Tools Library (xdTools.dll)
 // Contents: Declaration module for xdTokens utility class

//

// -----
 // Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----
 //

#ifndef _INC_XDTOKENS_H_
 #define _INC_XDTOKENS_H_

#if _MSC_VER >= 1000

#pragma once

#endif // _MSC_VER >= 1000

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h> // X:drive Tools Related

#define XD_MAX_TOKENS 1024

//

// -----
 // xdTokens
 // This class is a big worker class. its used to parse strings into
 // tokens or substrings. Strings are parsed by supplying a string of
 // characters which will be used to parse out the string.

//

class XDTOOLS_PUBLIC xdTokens

{

public:

xdTokens(LPCTSTR pTokens = NULL);

~xdTokens();

//

// Public Interface

//

public:

int Parse(int iNumToParse, LPCTSTR pString, LPCTSTR pTokens=NULL);

int Parse(LPCTSTR pString, LPCTSTR pTokens=NULL);

LPCTSTR operator[](int iIndex);

//

// Private Members

//

private:

LPCTSTR *m_pTok;
 int m_iNumParsed;
 LPTSTR m_szWorkString;
 LPTSTR m_szTokens;
 LPTSTR m_pWorkString;

}; // End of xdTokens

#endif // _INC_XDTOKENS_H_

//

// Module: xdTools.h

// Subsystem: X:drive Tools Library (xdTools.dll)
 // Contents: Main header file for the xdTools library

//

// -----
 // Copyright (c) 1999 by X:drive(tm), Inc.
 // Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
 // All rights reserved.

//

// -----
 //

#ifndef _INC_XDTOOLS_H_
 #define _INC_XDTOOLS_H_

#if _MSC_VER >= 1000
 #pragma once
 #endif // _MSC_VER >= 1000

#include <xdGlobals.h> // X:drive system wide globals
 #ifdef _VXD_SOURCE_
 #include <xdCString.h>
 #endif
 #pragma warning (disable : 4100)
 #pragma warning (disable : 4201)

//

// The following code block will insure the proper resolution of any
 // API functions (and classes) which are exposed from the XDTOOLS library.
 // When compiling the XDTOOLS library source code, make sure that the
 // following #define is defined in the project settings (both debug & release).
 // This will cause any classes and/or API functions defined as to
 // be exported to the LIB file. If you are USING the library by linking to
 // the XDTOOLS.LIB or XDTOOLS.D.LIB import libraries, then ignore the
 // following #define's for

//

#ifdef _XDTOOLS_SOURCE_
 #define XDTOOLS_PUBLIC __declspec(dllexport)
 #else
 #define XDTOOLS_PUBLIC // __declspec(dllimport)
 #endif // _XDTOOLS_SOURCE_

//

// If we are debugging & we trap an exception, we will display it
 // in a message box, otherwise in release mode, we wont.

//

#ifdef _DEBUG
 #define XDTRACE(x) AfxMessageBox(x)
 #else
 #define XDTRACE(x) TRACE0(x)
 #endif

//

// XDDATE API (Date Functions)

//

XDTOOLS_PUBLIC int XDDATE_MonthNum (LPTSTR szMonth);

//

// XDSTR API (String Functions)

//

```

XDTOOLS_PUBLIC LPTSTR XDSTR_Squish ( LPTSTR p );
XDTOOLS_PUBLIC LPTSTR XDSTR_StripChar ( LPTSTR p, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_DirSlashAdd ( LPTSTR sz, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_DirSlashRemove ( LPTSTR sz, TCHAR c );
XDTOOLS_PUBLIC LPTSTR XDSTR_TrimRight ( LPTSTR );
XDTOOLS_PUBLIC LPTSTR XDSTR_TrimLeft ( LPTSTR );
XDTOOLS_PUBLIC LPTSTR XDSTR_Trim ( LPTSTR );
XDTOOLS_PUBLIC BOOL XDAPI_CreatePath ( LPCTSTR ); // calls CreateDirectory() to make a path.

```

```

// -----
// Stuff for message boxes
//
#ifdef _VXD_SOURCE_
    int XDTOOLS_PUBLIC XD_MSG ( LPCTSTR szText, UINT uiMsgFlags );
    int XDTOOLS_PUBLIC XD_QUESTION ( LPCTSTR szText, UINT uiMsgFlags );
    LPCTSTR XDTOOLS_PUBLIC XD_TEXT ( HINSTANCE h, UINT uiResId ); // LOADS A
RESOURCE!
    BOOL XD_DoHelp ( LPHELPINFO );
    void XD_DoHelpContext ( CWnd* );
#endif

```

```

//
// the calling object needs to supply the resource
// handle for loading the string. So set up a stupid macro
// that will automatically supply this!
//
#define XD_LOADSTRING(x) XD_TEXT(AfxGetResourceHandle(),(x))

```

```

//
// DEBUGGING STUFF
//
#define CATCH_MSG_T("Caught Exception in File %s, Line %d\n\n")
#ifdef _VXD_SOURCE_
    #define XDCATCH dprintf(CATCH_MSG, T(__FILE__), __LINE__)
#else
    #define XDCATCH { CString s; s.Format(CATCH_MSG, T(__FILE__), __LINE__);
AfxMessageBox(s); }
#endif

```

```

//
// Ring 0 File I/O
//
#ifdef _VXD_SOURCE_
#define GENERIC_READ (0x80000000) /* from WINNT.H */
#define GENERIC_WRITE (0x40000000) /* from WINNT.H */
#define CREATE_NEW 1
#define CREATE_ALWAYS 2
#define OPEN_EXISTING 3
#define OPEN_ALWAYS 4
#define TRUNCATE_EXISTING 5
#define FILE_SHARE_READ 0x00000001
#define FILE_SHARE_WRITE 0x00000002
#define FILE_SHARE_DELETE 0x00000004 // not supported

```

```

HANDLE CreateFile ( LPCTSTR lpFileName, // pointer to name of the file
    DWORD dwDesiredAccess, // access (read-write) mode
    DWORD dwShareMode, // share mode
    void* lpSecAtt, // pointer to security
    attributes
    DWORD dwCreateFlags, // how to create
    DWORD dwFlagsAndAttributes, // file attributes
    HANDLE);

```

```

BOOL CloseHandle ( HANDLE hFile );
BOOL ReadFile ( HANDLE hFile,          // handle of file to read
               void* lpBuffer,          // pointer to buffer that receives data
               DWORD nNumberOfBytesToRead, // number of bytes to read
               DWORD* lpNumberOfBytesRead, // pointer to number of bytes read
               void* lpOverlapped);      // pointer to structure for data

BOOL ReadFileLine ( HANDLE hFile,          // handle of file to read
                   BYTE* lpBuffer,          // pointer to buffer that receives
data                                     data
                   DWORD dwBytesToRead,      // number of bytes to read
                   DWORD* dwBytesRead,        // pointer to number of bytes read
                   DWORD* dwOffset);          // pointer to structure for data

BOOL WriteFile ( HANDLE hFile, LPCTSTR lpBuffer, DWORD dwBytesToWrite,
                DWORD* pBytesWritten, void* p);

DWORD      GetFileSize ( HANDLE hFile, DWORD* pdwHigh );
#endif

#endif // !defined(_INC_XDTOOLS_H_)

```



```

//
// Module: xdEngine.h
// Subsystem: X:drive Client Engine (xdEngine.dll)
// Contents: Main include file for the xdEngine subsystem
//
// -----
// Copyright (c) 1999 by X:drive(tm), Inc.
// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.
// All rights reserved.
// -----
//
// #ifndef _INC_XDRIVE_ENGINE_H_
// #define _INC_XDRIVE_ENGINE_H_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#pragma warning (disable : 4100)
#pragma warning (disable : 4201)

#ifdef _XDENGINE_SOURCE_
#define XDAPI_PUBLIC __declspec( dllexport )
#else
#define XDAPI_PUBLIC // __declspec( dllimport )
#endif // _XDENGINE_SOURCE_

#pragma pack(1) // byte pack this thing!

#include <xdGlobals.h>

// -----
// XD_DIRENTRY - directory listing item
//
// The following structure is used to hold an object in the file listing
// file. Xdrive will generate the file list for the directory and store it
// in the cache directory. That file will contain
// a list of record structures of this type. The .mnd file is generated
// based upon the FTP server specific format in the .idx file in the same
// cache directory.
//
typedef struct _xd_direntry_
{
    USHORT                cb;                // class size, MUST BE FIRST!!!!
    DWORD                 dwFileAttributes;
    FILETIME              ftCreationTime;
    FILETIME              ftLastAccessTime;
    FILETIME              ftLastWriteTime;
    DWORD                 nFileSizeHigh;
    DWORD                 nFileSizeLow;
    TCHAR                 cFileName[ XD_LEN_512 ];
    TCHAR                 m_szObPerms [ XD_LEN_32 + 1 ];
    BYTE                  m_bObOwnerPerms[4];
    BYTE                  m_bObGroupPerms[4];
    BYTE                  m_bObWorldPerms[4];
} XD_DIRENTRY, * LPXD_DIRENTRY;

#pragma pack()

```

```

//
// Return codes
//
typedef UINT    XD_RETCODE;

#define XD_SUCCESS                (int)0
#define XD_CANCEL                 (int)1
#define XD_ERR_CONNECTFAILED     (int)2           // socket connect failed
#define XD_ERR_LOGINFAILED       (int)3           // bad username/pwd
#define XD_ERR_CONNECTREFUSED    (int)5           // socket connect refused
#define XD_ERR_CANTRESOLVEHOST   (int)6           // cant resolve host
#define XD_ERR_SERVERUPGRADING   (int)7           // upgrading our servers

#define XD_ERR_OTHER              (int)-1

//
// The following constants are used in the notification structure.
//
typedef enum
{
    XD_NOTIFY_IDLE                = 0,           // nothing happening here
    XD_NOTIFY_STATUS_MSG          = 1000,        // status msg
    XD_NOTIFY_XFERDATA_DN         = 1001,        // downloading
    XD_NOTIFY_XFERDATA_UP         = 1002,        // uploading
    XD_NOTIFY_QUOTA                = 1003,        // Update the quota
    XD_NOTIFY_START                = 1004,        // Start an operation
    XD_NOTIFY_STOP                 = 1005        // Stop an operation
} XD_NOTIFY_CODE;

// -----
// XD_NOTIFY - This is our notification structure. The http engine
// will use this structure to pass status information back to the
// invoking method.
//
#pragma pack(1)

typedef struct _xd_notification_
{
    int                m_iNotifyType;
    TCHAR              m_szMessage [ 1024 + sizeof(TCHAR) ];

    //
    // used for send/receive
    //
    ULONG              m_dwStartTime;           // GetTickCount()/1000
    ULONG              m_dwCurrentTime;         // GetTickCount()/1000

    DWORD              m_dwCurrentBytes;
    DWORD              m_dwTotalBytes;

    TCHAR              m_szLocalFileName [ MAX_PATH + sizeof(TCHAR) ];
    TCHAR              m_szRemoteFileName [ MAX_PATH + sizeof(TCHAR) ];
} XD_NOTIFY, *LPXD_NOTIFY;
#pragma pack()
#define XD_NOTIFY_MAX    50

#endif // _INC_XDRIVE_ENGINE_H_

```

//

// **Module: tdimsgtbl.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: TDI Error table.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by River Front Software

// All rights reserved.

// -----

// -----

//

#ifndef __TDIMSGTBL_H

#define __TDIMSGTBL_H

typedef struct

```
{
    TDI_STATUS    Status;
    int           WinStatus;
    char          *szMsg;
} INETTDIMSG;
```

INETTDIMSG TdiMsgTbl[] =

```
{
    {TDI_SUCCESS, ERROR_SUCCESS, "TDI Success"},
    {TDI_NO_RESOURCES, ERROR_BAD_COMMAND, "No resources."},
    {TDI_ADDR_IN_USE, ERROR_BAD_COMMAND, "Address already in use."},
    {TDI_BAD_ADDR, ERROR_BAD_COMMAND, "Address given is bad."},
    {TDI_NO_FREE_ADDR, ERROR_BAD_COMMAND, "No addresses available."},
    {TDI_ADDR_INVALID, ERROR_BAD_COMMAND, "Address object is invalid."},
    {TDI_ADDR_DELETED, ERROR_BAD_COMMAND, "Address object was deleted."},
    {TDI_BUFFER_OVERFLOW, ERROR_BAD_COMMAND, "Buffer overflowed."},
    {TDI_BAD_EVENT_TYPE, ERROR_BAD_COMMAND, "Bad event type."},
    {TDI_BAD_OPTION, ERROR_BAD_COMMAND, "Bad option or length."},
    {TDI_CONN_REFUSED, ERROR_BAD_COMMAND, "Connection was refused."},
    {TDI_INVALID_CONNECTION, ERROR_BAD_COMMAND, "Invalid connection."},
    {TDI_ALREADY_ASSOCIATED, ERROR_BAD_COMMAND, "Connection already associated."},
    {TDI_NOT_ASSOCIATED, ERROR_BAD_COMMAND, "Connection not associated."},
    {TDI_CONNECTION_ACTIVE, ERROR_BAD_COMMAND, "Connection is still active."},
    {TDI_CONNECTION_ABORTED, ERROR_BAD_COMMAND, "Connection was aborted."},
    {TDI_CONNECTION_RESET, ERROR_BAD_COMMAND, "Connection was reset."},
    {TDI_TIMED_OUT, ERROR_BAD_COMMAND, "Connection timed out."},
    {TDI_GRACEFUL_DISC, ERROR_BAD_COMMAND, "Received a graceful disconnect."},
    {TDI_NOT_ACCEPTED, ERROR_BAD_COMMAND, "Data not accepted."},
    {TDI_MORE_PROCESSING, ERROR_BAD_COMMAND, "More processing required."},
    {TDI_INVALID_STATE, ERROR_BAD_COMMAND, "TCB in an invalid state."},
    {TDI_INVALID_PARAMETER, ERROR_BAD_COMMAND, "An invalid parameter."},
    {TDI_DEST_NET_UNREACH, ERROR_BAD_COMMAND, "Destination net is unreachable."},
    {TDI_DEST_HOST_UNREACH, ERROR_BAD_COMMAND, "Dest. host is unreachable."},
    {TDI_DEST_UNREACHABLE, ERROR_BAD_COMMAND, "Dest. is unreachable."},
    {TDI_DEST_PROT_UNREACH, ERROR_BAD_COMMAND, "Destination protocol is unreachable."},
    {TDI_DEST_PORT_UNREACH, ERROR_BAD_COMMAND, "Dest. port is unreachable."},
    {TDI_INVALID_QUERY, ERROR_BAD_COMMAND, "Invalid query type specified."},
    {TDI_REQ_ABORTED, ERROR_BAD_COMMAND, "Request was aborted for some reason."},
    {TDI_BUFFER_TOO_SMALL, ERROR_BAD_COMMAND, "Buffer was too small."},
    {TDI_CANCELLED, ERROR_BAD_COMMAND, "The request was cancelled."},
    {TDI_BUFFER_TOO_BIG, ERROR_BAD_COMMAND, "Invalid request."},
    {ERROR_SEM_TIMEOUT, ERROR_SEM_TIMEOUT, "Timed out."},
    {TDI_PENDING, ERROR_BAD_COMMAND, "Pending"}
}
```

};

#endif

//

// **Module: tdisock.h**

// Subsystem: X:drive Client Engine (xdEngine.dll)

// Contents: TDI Socket header file.

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by River Front Software

// All rights reserved.

//

// -----

//

#ifndef __TDISOCK_H

#define __TDISOCK_H

#define TDISOCK_TIMEOUT 15000

#define WSADESCRIPTION_LEN 256

#define WSASYS_STATUS_LEN 128

typedef short SHORT;

typedef unsigned short USHORT;

typedef unsigned short ushort;

typedef unsigned int uint;

typedef unsigned long ulong;

typedef unsigned long ULONG;

typedef void (*CTEReqCmpltRtn)(void *Context, long FinalStatus, unsigned int ByteCount);

typedef unsigned char uchar;

typedef struct WSADATA {

WORD wVersion;

WORD wHighVersion;

char szDescription[WSADESCRIPTION_LEN+1];

char szSystemStatus[WSASYS_STATUS_LEN+1];

unsigned short iMaxSockets;

unsigned short iMaxUdpDg;

char FAR * lpVendorInfo;

} WSADATA;

typedef WSADATA FAR *LPWSADATA;

#define USE_NDIS 1

#include <vtoolscp.h>

#include <crtl.h>

#undef USE_NDIS

#include <tdi.h>

#include <vxdsvc.h>

#include <tdivxd.h>

#include <tdistat.h>

#undef VTDI_Device_ID

#include <vtidi.h>

#define MAKELONG(a, b) ((LONG)(((WORD)(a)) | ((DWORD)((WORD)(b))) << 16))

#define LOWORD(l) ((WORD)(l))

```

#define HIWORD(l)      ((WORD)(((DWORD)(l) >> 16) & 0xFFFF))
#define LOBYTE(w)      ((BYTE)(w))
#define HIBYTE(w)      ((BYTE)(((WORD)(w) >> 8) & 0xFF))

/*
 * Structures returned by network data base library, taken from the
 * BSD file netdb.h. All addresses are supplied in host order, and
 * returned in network order (suitable for use in system calls).
 */

struct hostent {
    char FAR * h_name;          /* official name of host */
    char FAR * FAR * h_aliases; /* alias list */
    short h_addrtype;          /* host address type */
    short h_length;            /* length of address */
    char FAR * FAR * h_addr_list; /* list of addresses */
#define h_addr h_addr_list[0]    /* address, for backward compat */
};

/***** Wait for semaphore flags */
#define WAIT_SEMA_FLAGS 0 //BLOCK_SVC_INTS | BLOCK_POLL

/***** Macro to call wait on semaphore function */
#define SEMAPHORE_WAIT( hSem, nTimeout ) \
    WaitOnSemaphore( s, hSem, #hSem, nTimeout )

/***** Checks for valid TDI status */
#define TDI_CHECKSTATUS(s) if ( (s) != TDI_SUCCESS ) \
    { \
        \
        errdebug( DBG_log("ERROR - File: %s \
        \
        Line:%d TDI [%d] - %s\n", \
        \
        MapTdiToString(s)); ); \
        \
        goto Exit; \
    } \
    \

/***** Destroys a semaphore */
#define SEMAPHORE_SAFE_DESTROY(hSem) \
    if (hSem) \
    { \
        vbsdebug( DBG_log("Destroy Semaphore %s", #hSem); ); \
        UtilSemDestroy(hSem); \
        hSem = 0; \
    } \

/***** Signals a semaphore */
#define SEMAPHORE_SAFE_SIGNAL(hSem) \
    if (hSem) \
    { \
        vbsdebug( DBG_log("**** Signal Semaphore %s", #hSem); ); \
        vbsdebug( DBG_log_hex_long( hSem ); ); \
        Signal_Semaphore_No_Switch( hSem ); \
    } \
    else \
    { \
        vbsdebug( DBG_log("**** NO SEMAPHORE TO SIGNAL %s", #hSem); ); \
    } \

```

```

/*
 * Basic system type definitions, taken from the BSD file sys/types.h.
 */
typedef unsigned char  u_char;
typedef unsigned short u_short;
typedef unsigned int    u_int;
typedef unsigned long   u_long;

/*
 * Constants and structures defined by the internet system,
 * Per RFC 790, September 1981, taken from the BSD file netinet/in.h.
 */

/*
 * Protocols
 */
#define IPPROTO_IP      0      /* dummy for IP */
#define IPPROTO_ICMP    1      /* control message protocol */
#define IPPROTO_IGMP    2      /* internet group management protocol */
#define IPPROTO_GGP     3      /* gateway^2 (deprecated) */
#define IPPROTO_TCP     6      /* tcp */
#define IPPROTO_PUP     12     /* pup */
#define IPPROTO_UDP     17     /* user datagram protocol */
#define IPPROTO_IDP     22     /* xns idp */
#define IPPROTO_ND      77     /* UNOFFICIAL net disk proto */

#define IPPROTO_RAW     255     /* raw IP packet */
#define IPPROTO_MAX     256

/*
 * Port/socket numbers: network standard functions
 */
#define IPPORT_ECHO      7
#define IPPORT_DISCARD  9
#define IPPORT_SYSTAT   11
#define IPPORT_DAYTIME  13
#define IPPORT_NETSTAT  15
#define IPPORT_FTP      21
#define IPPORT_TELNET   23
#define IPPORT_SMTP     25
#define IPPORT_TIMESERVER 37
#define IPPORT_NAMESERVER 42
#define IPPORT_WHOIS    43
#define IPPORT_MTP      57

/*
 * Port/socket numbers: host specific functions
 */
#define IPPORT_TFTP      69
#define IPPORT_RJE      77
#define IPPORT_FINGER    79
#define IPPORT_TTYLINK  87
#define IPPORT_SUPDUP    95

/*
 * UNIX TCP sockets
 */
#define IPPORT_EXECSERVER 512

```

```

#define IPPORT_LOGINSERVER  513
#define IPPORT_CMDSERVER    514
#define IPPORT_EFSSERVER    520

/*
 * UNIX UDP sockets
 */
#define IPPORT_BIFFUDP      512
#define IPPORT_WHOSERVER    513
#define IPPORT_ROUTESERVER  520
                        /* 520+1 also used */

/*
 * Ports < IPPORT_RESERVED are reserved for
 * privileged processes (e.g. root).
 */
#define IPPORT_RESERVED    1024

/*
 * Link numbers
 */
#define IMPLINK_IP          155
#define IMPLINK_LOWEXPER    156
#define IMPLINK_HIGHEXPER   158

/*
 * Internet address (old style... should be updated)
 */
struct in_addr {
    union {
        struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b;
        struct { u_short s_w1,s_w2; } S_un_w;
        u_long S_addr;
    } S_un;
#define s_addr S_un.S_addr
                        /* can be used for most tcp & ip code */
#define s_host S_un.S_un_b.s_b2
                        /* host on imp */
#define s_net S_un.S_un_b.s_b1
                        /* network */
#define s_imp S_un.S_un_w.s_w2
                        /* imp */
#define s_impno S_un.S_un_b.s_b4
                        /* imp # */
#define s_lh S_un.S_un_b.s_b3
                        /* logical host */
};

#define htons(host) ( (((host) & 0xff) << 8) | ((host) >> 8) )
#define htonl( ULONG hostlong );

/*
 * Definitions of bits in internet address integers.
 * On subnets, the decomposition of addresses to host and net parts
 * is done according to subnet mask, not the masks here.
 */
#define IN_CLASSA(i)      (((long)(i) & 0x80000000) == 0)
#define IN_CLASSA_NET      0xff000000
#define IN_CLASSA_NSHIFT   24
#define IN_CLASSA_HOST     0x00ffffff
#define IN_CLASSA_MAX      128

```



```
#define IN_CLASSB(i)      (((long)(i) & 0xc0000000) == 0x80000000)
#define IN_CLASSB_NET      0xffff0000
```

```
// end first 30 pages aj
```

```

        int iMax = i;
        CString* pArray = new CString[iMax];
        i=0;
        while (r1.RegEnumKey(i++,szVal,dwCnt))
            pArray[i-1] = szVal;
        r1.RegClose();
        for (i=0; i<iMax; i++)
        {
            CString str = pArray[i];
            CString strTmp;
            strTmp.Format(_T("%s\\%s"), (LPCTSTR)szSubKey, (LPCTSTR)str);
            r1.RegDeleteKey(hHive,strTmp);
        }
        delete[] pArray;
    }

    //
    // then Delete the key
    //
    m_lRetCode = ::RegDeleteKey ( hHive, szSubKey );
#endif

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    //
    // bOK is TRUE if ERROR_SUCCESS was returned
    //
    bOK = (ERROR_SUCCESS == m_lRetCode);

    return bOK;
} // End of RegDelete()

// -----
// Method: RegClose()
// Purpose: the the registry is open, close it.
//
BOOL xdRegistry::RegClose ( )
{
    BOOL bOK = TRUE;

#ifdef _VXD_SOURCE_
    try
    {
#endif
        if ( m_hKey != NULL )
            ::RegCloseKey ( m_hKey );

```

```

    #ifndef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    //
    // unconditionally null the key
    //
    m_hKey = NULL;

    return bOK;
} // End of RegClose()

// -----
// Method: RegEnumStr()
// Purpose: enumerates subkeys for a key. i is the index to get
//
BOOL xdRegistry::RegEnumStr ( int i, LPCTSTR szValue, UINT uiLenWithNull )
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiLenWithNull;
    LPBYTE pValue = (LPBYTE) szValue;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifndef _VXD_SOURCE_
    try
    {
#endif
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiLenWithNull );

#ifdef _VXD_SOURCE_
        m_lRetCode = ::RegEnumKey ( m_hKey,
                                     dwIdx,
                                     (LPTSTR)pValue,
                                     dwSize);
        // hive/key
        // index
        // key
        // the size of the

        // buffer
        #else
        #ifdef _UNICODE
            CString sTmp;
            TCHAR szBuf = (BYTE*)sTmp.GetBuffer(512);
            m_lRetCode = ::RegEnumKeyA (m_hKey,
                                         dwIdx,
                                         (char*)buf,
                                         dwSize);
            // hive/key
            // index of the
            // key name will
            // the size of the buffer

            CString fred(buf);

```

```

        _tcscpy((LPTSTR)szValue,fred);
    #else
        m_lRetCode = ::RegEnumKey (    m_hKey,
                                        // hive/key
                                        dwIdx,           // index
                                        (LPTSTR)pValue,   // key
                                        dwSize);          // the size of the
    #endif
    #endif
    #endif

    bOK = (ERROR_SUCCESS == m_lRetCode);
    if (bOK != FALSE)
    {
        //
        // terminate the string...ensure that we dont go past
        // the max lenh of the string!
        //
        ((LPTSTR)szValue) [ min(dwSize,uiLenWithNull) ] = 0;
    }

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegEnumStr()

// -----
// Method: RegGetStr()
// Purpose: retrieves a string value from the registry. NOTE: The length
//          of the string MUST include space for the NULL terminator since
//          this character IS read from the registry. So, if you want to
//          read 'ABCD' from the registry, supply a uiLenWithNull of five(5).
//
BOOL xdRegistry::RegGetStr ( LPCTSTR szName, LPCTSTR szValue, UINT uiLenWithNull )
{
    BOOL bOK = TRUE;
    DWORD dwType = 0;
    DWORD dwSize = (DWORD) uiLenWithNull;
    LPBYTE pValue = (LPBYTE) szValue;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiLenWithNull );

```

```

#ifdef _VXD_SOURCE_
    m_lRetCode = ::RegQueryValueEx (m_hKey,
        // value name
        // reserved
        // the REG_* type
        // pointer to the storage area
        // # to fetch (WITH NULL)
#else
    #ifdef _UNICODE
        char sShort[512];
        char sDefault[512];
        char buf[512];
        BOOL b;
        *sDefault = *sShort=0;
        WideCharToMultiByte (CP_ACP, 0, szName, -1, sShort, 512, sDefault, &b );
        m_lRetCode = ::RegQueryValueExA (m_hKey,
            // value name
            // reserved
            // the REG_* type
            // pointer to the storage area
            // # to fetch (WITH NULL)
            CString fred(buf);
            _tcsncpy((LPTSTR)szValue,fred);
    #else
        m_lRetCode = ::RegQueryValueEx (m_hKey,
            // hive/key
            szName,
            // value
            0,
            &dwType,
            // the
            pValue,
            //
            &dwSize);
            // # to
    #endif
#endif

bOK = (ERROR_SUCCESS == m_lRetCode);
if ( bOK == TRUE )
{
    //
    // make sure that it was a string value which was returned.
    // If not, Delete the entry so we can regen it as a string
    //
    if (REG_SZ != dwType)
        ::RegDeleteValue ( m_hKey, (LPTSTR)szName );

    //
    // terminate the string...ensure that we dont go past
    // the max lenth of the string!

```

```

        //
        ((LPTSTR)szValue) [ min(dwSize,uiLenWithNull) ] = 0;
    }
#endif _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegGetStr()

// -----
// Method: RegPutStr()
// Purpose: write the information to the registry (write the NULL TOO).
//
BOOL xdRegistry::RegPutStr ( LPCTSTR szName, LPCTSTR szValue )
{
    BOOL bOK = TRUE;

    //
    // Make sure that the registry is open
    //
    if(m_hKey == NULL)
        return FALSE;

#ifndef _VXD_SOURCE_
    try
    {
#endif

#ifdef _VXD_SOURCE_
        //
        // move everything into a temp buffer so that we can ensure
        // the existence of a NULL byte on the end of the string
        //
        CString sTmp;
        LPTSTR szBuf = sTmp.GetBuffer(512);
        memset ( szBuf, 0, 512 );
        memcpy ( szBuf, szValue, min(sTmp.GetAllocLength()-1,strlen(szValue)) );

        //
        // remember....always write the NULL byte too!
        //
        UINT uiLenWithNull = strlen(szBuf) + 1;
        m_lRetCode = ::RegSetValueEx ( m_hKey, (LPTSTR)szName, 0, REG_SZ,
                                     (LPBYTE)szBuf,
                                     uiLenWithNull );
    #else
        #ifdef _UNICODE
            char sShort[512];
            char sShortVal[512];
            char sDefault[512];
            BOOL b;
            *sDefault = *sShort=0;
            WideCharToMultiByte ( CP_ACP, 0, szName, -1, sShort, 512, sDefault, &b );
            WideCharToMultiByte ( CP_ACP, 0, szValue, -1, sShortVal, 512, sDefault, &b );
            m_lRetCode = ::RegSetValueExA ( m_hKey, sShort, 0, REG_SZ,

```

```

        strlen(sShortVal)+1 );
        #else
            CString sTmp;
            LPTSTR szBuf = (LPTSTR)sTmp.GetBuffer(1024);
            memset ( szBuf, 0, 1024 );
            memcpy ( szBuf, szValue, min(1023, _tcslen(szValue))*sizeof(TCHAR) );
            szBuf[_tcslen(szValue)] = 0;

            //
            // remember...always write the NULL byte too!
            //
            UINT    uiLenWithNull = _tcslen(szBuf) + 1;

            m_lRetCode = ::RegSetValueEx ( m_hKey, szName, 0, REG_SZ,
                (LPBYTE) szBuf,
                uiLenWithNull );
            #endif
        #endif

        bOK = (ERROR_SUCCESS == m_lRetCode);

    #ifndef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
    #endif

    return bOK;
} // End of RegPutStr()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. there are various
//           overloads for different types.
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, DWORD& dwValue)
{
    BOOL    bOK = TRUE;
    CString sTmp;
    LPTSTR  szBuf = sTmp.GetBuffer(XD_LEN_64);
    memset ( szBuf, 0, XD_LEN_64 );
    DWORD   dwType = 0;
    DWORD    dwSize = XD_LEN_64-1;

    //
    // Make sure that the registry is open
    //
    if (m_hKey == NULL)
        return FALSE;

    #ifndef _VXD_SOURCE_
    try
    {
    #endif

    #ifdef _VXD_SOURCE_
        bOK = RegGetStr ( sName, szBuf, sTmp.GetAllocLength()-1 );
        if ( bOK == TRUE )
    #endif

```

```

        dwValue = (DWORD)atol((LPTSTR)szBuf);
#else
    #ifdef _UNICODE
        char sShort[512];
        char sDefault[512];
        char bufTmp[512];
        BOOL b=0;
        *sDefault = *sShort=0;
        WideCharToMultiByte ( CP_ACP, 0, sName, -1, sShort, 512, sDefault, &b );
        m_lRetCode = ::RegQueryValueExA (m_hKey,                // hive/key
                                         sShort,
                                         // value name
                                         0,
                                         // reserved
                                         &dwType,
                                         // the REG_* type
                                         (LPBYTE)bufTmp,
                                         // pointer to the storage area
                                         &dwSize);
        // # to fetch (WITH NULL)
        bOK = (ERROR_SUCCESS == m_lRetCode);
        if ( bOK == TRUE )
        {
            if ( dwType == REG_SZ )
                dwValue = (DWORD)atol(bufTmp);
        }
    #else
        m_lRetCode = ::RegQueryValueEx (    m_hKey,
                                           sName,
                                           0,
                                           &dwType,
                                           (BYTE*)szBuf,
                                           &dwSize );

        bOK = (ERROR_SUCCESS == m_lRetCode);
        if ( bOK == TRUE )
        {
            if ( dwType == REG_SZ )
                dwValue = (DWORD)_ttoi((LPTSTR)szBuf);
            if ( dwType == REG_DWORD )
                dwValue = * ((DWORD*)szBuf);
        }
    #endif
#endif

    #ifndef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. UINT version
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, UINT& uiValue)
{

```

```

        DWORD      dwValue = uiValue;
        BOOL  bOK = RegGetNum(sName,dwValue);

        uiValue = (UINT) dwValue;

        return bOK;
    } // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. BOOL version
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, BOOL& bValue)
{
    DWORD      dwValue = bValue;
    BOOL  bOK = RegGetNum(sName,dwValue);

    bValue = (BOOL) dwValue;

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegGetNum()
// Purpose: Retrieves a number from the registry. WORD VERSION.
//
BOOL xdRegistry::RegGetNum(LPCTSTR sName, WORD& wValue)
{
    DWORD      dwValue = wValue;
    BOOL  bOK = RegGetNum(sName,dwValue);

    wValue = (WORD) dwValue;

    return bOK;
} // End of RegGetNum()

// -----
// Method: RegPutNum()
// Purpose: writes a numeric value to the registry.
//
BOOL xdRegistry::RegPutNum(LPCTSTR sName, DWORD dwValue)
{
    BOOL  bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif
#ifdef _VXD_SOURCE_
        CString sTmp;
        BYTE* szBuf = (BYTE*)sTmp.GetBuffer(132);
        sprintf( (LPTSTR)szBuf, _T("%lu"), dwValue);
        UINT uiLenWithNull = strlen((LPTSTR)szBuf) + 1; // ADD THE NULL!!!!!!
        m_lRetCode = ::RegSetValueEx ( m_hKey, (LPTSTR)sName,

```



```

    uiLenWithNull );
        bOK = (ERROR_SUCCESS == m_lRetCode);
    #else
        #ifdef _UNICODE
            char sShort[512];
            char sDefault[512];
            BOOL b;
            *sDefault = *sShort=0;
            WideCharToMultiByte ( CP_ACP, 0, sName, -1, sShort, 512, sDefault, &b );
            sprintf( sDefault, "%lu", dwValue );
            m_lRetCode = ::RegSetValueExA (m_hKey, sShort, 0, REG_SZ,
                (LPBYTE)sDefault,
                strlen(sDefault)+1 );
        #else
            CString sTmp;
            LPTSTR szBuf = sTmp.GetBuffer(XD_LEN_64);
            wsprintf( (LPTSTR)szBuf, _T("%lu"), dwValue);
            UINT uiLenWithNull = _tcslen((LPTSTR)szBuf) + 1; // ADD THE NULL!!!!!!
            m_lRetCode = ::RegSetValueEx ( m_hKey,
                sName,
                0,
                REG_SZ,
                (BYTE*)szBuf,
                uiLenWithNull);
        #endif
    #endif

    bOK = (ERROR_SUCCESS == m_lRetCode);

    #ifndef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
    #endif

    return bOK;
} // End of RegPutNum()

// -----
// Method: RegDeleteValue()
// Purpose:
//
BOOL xdRegistry::RegDeleteValue ( LPCTSTR szValue )
{
    BOOL bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

    #ifndef _VXD_SOURCE_
    try
    {
    #endif

    m_lRetCode = ::RegDeleteValue ( m_hKey, (LPTSTR)szValue );

```

```

        bOK = (ERROR_SUCCESS == m_lRetCode);

#ifdef _VXD_SOURCE_
    }
    catch(...)
    {
        XDCATCH;
        bOK = FALSE;
    }
#endif

    return bOK;
} // End of RegDeleteValue()

// -----
// Method: RegEnumVal()
// Purpose: enumerates values for a key. i is the index to get
//
BOOL xdRegistry::RegEnumVal ( int i, LPCTSTR szValueName, UINT uiNameLenWithNull,
                             LPCTSTR szValueData, UINT
uiDataLenWithNull)
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiNameLenWithNull;
    DWORD dwDataSize = (DWORD) uiDataLenWithNull;
    LPBYTE pValue = (LPBYTE) szValueName;
    LPBYTE pDataValue = (LPBYTE) szValueData;

    //
    // make sure the key is open
    //
    if (m_hKey == NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif

        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiNameLenWithNull );
        memset ( pDataValue, 0, uiDataLenWithNull );

#ifdef _VXD_SOURCE_
        m_lRetCode = ::RegEnumValue(m_hKey,
of the value to get          dwIdx,          // index
valuenam will go here      (LPCTSTR)pValue,      //
size of the buffer         &dwSize,          // the
// reserved,               0,
address of type code        NULL,              //
                             pDataValue,
                             &dwDataSize);
#else
        m_lRetCode = ::RegEnumValue(m_hKey,
// hive/key

```

of the value to get
 valuenamewill go here
 size of the buffer

// reserved,

address of type code

#endif

```

bOK = (ERROR_SUCCESS == m_lRetCode);
if ( bOK == TRUE )
{
    //
    // terminate the string...ensure that we dont go past
    // the max lenh of the string!
    //
    ((LPTSTR)szValueName) [ min(dwSize,uiNameLenWithNull) ] = 0;
    ((LPTSTR)szValueData) [ min(dwDataSize,uiDataLenWithNull) ] = 0;
}

```

#ifndef _VXD_SOURCE_

```

}
catch(...)
{
    XDCATCH;
    bOK = FALSE;
}

```

#endif

```

    return bOK;
} // End of RegEnumVal()

```

```

// -----
// Method: RegPutBin()
// Purpose: write the information to the registry
//
BOOL xdRegistry::RegPutBin ( LPCTSTR szName, BYTE* pBuffer, UINT uiLength )
{

```

```

    BOOL bOK = TRUE;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

```

#ifndef _VXD_SOURCE_

```

try
{

```

#endif

```

    //
    // move everything into a temp buffer so that we can ensure
    // the existence of a NULL byte on the end of the string
    //
    CString sTmp;
    LPTSTR szBuf = sTmp.GetBuffer(132);
    memset ( szBuf, 0, 132 );

```

```

dwIdx,                // index
(LPTSTR)pValue,       //
&dwSize,              // the
0,
NULL,                 //
pDataValue,
&dwDataSize);

```

```

memcpy ( szBuf, pBuffer, min(sTmp.GetAllocLength()-1,uiLength) );

m_IRetCode = ::RegSetValueEx ( m_hKey,

(LPTSTR)szName,
0,
REG_BINARY,
(LPBYTE) szBuf,
uiLength );

bOK = (ERROR_SUCCESS == m_IRetCode);

#ifdef _VXD_SOURCE_
}
catch(...)
{
    XDCATCH;
    bOK = FALSE;
}
#endif

return bOK;
} // End of RegPutBin()

// -----
// Method: RegEnumKey()
// Purpose: enumerates values for a key. i is the index to get
//
BOOL xdRegistry::RegEnumKey ( int i, LPCTSTR szValueName, UINT uiNameLenWithNull)
{
    BOOL bOK = TRUE;
    DWORD dwIdx = i;
    DWORD dwSize = (DWORD) uiNameLenWithNull;
    LPBYTE pValue = (LPBYTE) szValueName;

    //
    // make sure the key is open
    //
    if (m_hKey==NULL)
        return FALSE;

#ifdef _VXD_SOURCE_
    try
    {
#endif
        //
        // initialize the string to be empty
        //
        memset ( pValue, 0, uiNameLenWithNull );

#ifdef _VXD_SOURCE_
        m_IRetCode = ::RegEnumKey(m_hKey,

// hive/key
dwIdx, // index of the
value to get
(LPTSTR)pValue, // valuenamewill
go here
dwSize); // the size of the
buffer
#else
        m_IRetCode = ::RegEnumKey(m_hKey,

// hive/key
dwIdx, // index of the
value to get
(LPTSTR)pValue, // valuenamewill
go here

```

```
buffer
#endif

dwSize); // the size of the

bOK = (ERROR_SUCCESS == m_lRetCode);
if (bOK==TRUE)
{
    //
    // terminate the string...ensure that we dont go past
    // the max lenth of the string!
    //
    ((LPTSTR)szValueName) [ min(dwSize,uiNameLenWithNull) ] = 0;
}

#ifdef _VXD_SOURCE_
}
catch(...)
{
    XDCATCH;
    bOK = FALSE;
}
#endif

return bOK;
} // End of RegEnumKey()
```

//

// **Module: xFileIO.cpp**

// Subsystem: X:drive Tools Library (xdTools.dll)

// Contents: Redefinitions for the FILE IO functions

//

// -----

// Copyright (c) 1999 by X:drive(tm), Inc.

// Portions Copyright (c) 1996-1999 by KnoWare(r), Inc.

// All rights reserved.

//

// -----

//

#include "stdafx.h"

#include <xdGlobals.h> // X:drive system wide globals

#include <xdTools.h>

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

#ifdef _VXD_SOURCE_

#include LOCKED_CODE_SEGMENT

#include LOCKED_DATA_SEGMENT

#endif

#ifdef _VXD_SOURCE_

//

// Function: CreateFile()

// Purpose: This API function maps the standard Win32 CreateFile function
// to the Ring-0 R0_OpenCreateFile() call.

// Returns: INVALID_HANDLE_VALUE - bad

// something else - good!

//

HANDLE CreateFile (LPCTSTR lpFileName, // pointer to name of the file

DWORD dwDesiredAccess, // access (read-write) mode

DWORD dwShareMode, // share mode

void* lpSecAtt, // pointer to security

attributes

DWORD dwCreateFlags, // how to create

DWORD dwFlagsAndAttributes, // file attributes

HANDLE)

{

HANDLE h = INVALID_HANDLE_VALUE;

WORD wError = 0;

WORD wMode = 0;

BYTE action = 0;

switch (dwDesiredAccess)

{

case GENERIC_READ:

wMode = OPEN_ACCESS_READONLY;

break;

case GENERIC_WRITE:

wMode = OPEN_ACCESS_WRITEONLY;

break;

default:

wMode = OPEN_ACCESS_READWRITE;

break;

}

```

//
// file sharing not supported!
//
wMode |= OPEN_SHARE_COMPATIBLE;

//
// Create Attributes
//
switch ( dwCreateFlags )
{
case CREATE_NEW: // create New file. fail if file exists
    action = ACTION_IFEXISTS_FAIL | ACTION_IFNOTEXISTS_CREATE;
    break;
case CREATE_ALWAYS: // create New file. overwrite if exists
    action = ACTION_IFEXISTS_TRUNCATE | ACTION_IFNOTEXISTS_CREATE;
    break;
case OPEN_EXISTING: // open file, fail if the file does not exists
    action = ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_FAIL;
    break;
case OPEN_ALWAYS: // open file. if !exists, create
    action = ACTION_IFEXISTS_OPEN | ACTION_IFNOTEXISTS_CREATE;
    break;
case TRUNCATE_EXISTING: // open&truncate file. fail if it does not exist
    action = ACTION_IFEXISTS_OPEN | ACTION_IFEXISTS_TRUNCATE |
ACTION_IFNOTEXISTS_FAIL;
    break;
}

h = R0_OpenCreateFile(1,(LPTSTR)lpFileName,wMode,
ATTR_NORMAL,action,R0_NO_CACHE,&wError,
&action);
return h;
} // End of CreateFile()

// -----
// Function: ReadFile()
// Purpose: This API function maps the standard Win32 ReadFile function
//          to the Ring-0 R0_ReadFile() call.
// Returns: TRUE - Good read
//          FALSE - Bad Read
//
BOOL ReadFile ( HANDLE hFile, void* lpBuffer, DWORD dwBytesToRead,
DWORD* pdwBytesRead, void* pdwOffset)
{
    WORD wError = 0;
    DWORD dwOffset = 0;

    if ( pdwOffset )
        dwOffset = *((DWORD*)pdwOffset);

    *pdwBytesRead = R0_ReadFile ( TRUE, hFile, lpBuffer, dwBytesToRead,
dwOffset, &wError );

    return ( wError == 0 );
} // End of ReadFile()

// -----
// Function: WriteFile()
// Purpose: This API function maps the standard Win32 WriteFile function
//          to the Ring-0 R0_WriteFile() call.
// Returns: TRUE - Good write
//          FALSE - Bad write

```

```

//
BOOL WriteFile ( HANDLE hFile, LPCTSTR lpBuffer, DWORD dwBytesToWrite,
                DWORD* pBytesWritten, void* p)
{
    WORD wError = 0;
    DWORD dwFilePos = R0_GetFileSize(hFile,&wError);
    *pBytesWritten = R0_WriteFile ( TRUE, hFile, (void*)lpBuffer, dwBytesToWrite,
                                   dwFilePos, &wError );

    return (wError == 0);
} // End of WriteFile()

// -----
// Function: CloseHandle()
// Purpose: This API function maps the standard Win32 CloseHandle function
//          to the Ring-0 R0_CloseFile() call.
// Returns: TRUE - success
//          FALSE - failure
//
BOOL CloseHandle ( HANDLE hFile )
{
    WORD wError = 0;
    return R0_CloseFile ( hFile, &wError );
} // End of CloseHandle()

// -----
// Function: GetFileSize()
// Purpose: This API function maps the standard Win32 GetFileSize function
//          to the Ring-0 R0_GetFileSize() call.
// Returns: TRUE - success
//          FALSE - failure
//
DWORD GetFileSize ( HANDLE hFile, DWORD* pdwHigh )
{
    WORD wError = 0;
    return R0_GetFileSize ( hFile, &wError );
} // End of GetFileSize()

// -----
// Function: ReadFileLine()
// Purpose: This API function maps the standard Win32 ReadFile function
//          to the Ring-0 R0_ReadFile() call.
// Returns: TRUE - Good read
//          FALSE - Bad Read
//
BOOL ReadFileLine ( HANDLE hFile, BYTE* lpBuffer,
                   DWORD dwBytesToRead,
                   DWORD* pdwBytesRead,
                   DWORD* pdwOffset )
{
    WORD wError = 0;
    DWORD dwOffset = 0;

    if ( pdwOffset )
        dwOffset = *((DWORD*)pdwOffset);

    //
    // Check for EOF
    //
    if ( dwOffset >= R0_GetFileSize(hFile,&wError) )
        return FALSE;

```



```

// *pdwBytesRead = R0_ReadFile ( TRUE, hFile, lpBuffer, dwBytesToRead,
//                               dwOffset, &wError );

memset ( lpBuffer, 0, dwBytesToRead );

int iTmpBytesRead = 1;
BOOL bFoundEOL = FALSE;
int i=0;
for ( i=0; (iTmpBytesRead != 0) && (i<dwBytesToRead) &&
      (wError == 0) && (bFoundEOL==FALSE); i++ )
{
    iTmpBytesRead = R0_ReadFile ( TRUE, hFile, &(lpBuffer[i]), 1, dwOffset+i, &wError );
    if ((iTmpBytesRead != 0) && (wError == 0))
    {
        if ( lpBuffer[i] == chNL )
            bFoundEOL = TRUE;
    }
}

*pdwBytesRead = i;

return ( wError == 0 );
} // End of ReadFileLine()
#endif

```